# 前　　言

作为数值计算、符号运算和图形处理等多种功能的强有力实现工具，近年来 MATLAB 这一强大的科学计算软件包得到了业界的广泛认可，并已深入到了各个行业的众多学科、在各大公司、科研机构、大学校园得到日益普及与广泛应用，其自身也因此得到了迅速发展，功能不断扩充，现已发展至 MATLAB 7。

## 本书特色

为了更好地推动 MATLAB 在各个行业、学科中的应用，在借鉴以往类似书籍经验并弥补其中不足的基础上，我们结合最新的 MATLAB 7 编撰了此书。全书力求从实用角度出发，通过大量典型的样例，对 MATLAB 7 的功能、操作及其相关应用进行详细论述。作为一本使读者对 MATLAB 快速上手和熟练掌握的书籍，全书紧密围绕精心设计的经典实例展开，而且这些实例通过了实际调试，以随书光盘的形式提供给读者。

本书全面、重点地介绍了下列 MATLAB 7 的新功能与特点：

- 代码效率分析
- M-Lint 编码分析
- 嵌套函数
- 匿名函数
- 直接从图形窗体生成 M 代码
- 文件 I/O 函数等

另外，MATLAB 与其他高级开发语言实现扩展编程，充分利用两者的优势是 MATLAB 的一个研究与应用热点，本书用了较多篇幅对其进行介绍。

## 本书导读

全书分 37 章进行展开，分为基础篇和实战篇。基础篇为前面的 34 章，主要讲述 MATLAB 的基本使用；实战篇为后面的 3 章，主要讲述 MATLAB 综合应用的实例。

全书内容概要如下：

- 第 1 章“基础入门”，介绍 MATLAB 发展历程、系统结构、工具箱、MATLAB 7/Simulink 6 最新特点、MATLAB 启动和退出，以及 MATLAB 基本特色。

  *本章重点讲述了 MATLAB 7/Simulink 6 特点，这些特点是 MATLAB 7 版本提供的新功能，这些新功能也将在后面章节重点进行讲述，熟悉和掌握这些新功能是掌握 MATLAB 7 的关键。*

- 第 2 章“MATLAB 桌面”，介绍 MATLAB 的主菜单及功能、命令窗口、工作空间、文件管理和帮助使用。

  *MATLAB 7 版本的交互式桌面更便于用户使用，熟悉和掌握这些交互式桌面的*

基本功能和基本特色，用户可以熟练地使用 MATLAB 完成相应的工作。

- 第 3 章“数组及其操作”，以及在此基础上展开的第 4 章“多维数组及其操作”，介绍了 MATLAB 中最核心的数据结构——数组及其各种基本操作。

  数组是 MATLAB 中各种变量存储和运算的通用数据结构，理解 MATLAB 中数组的结构并对其进行各种基本操作，是学习 MATLAB 应用和编程的第一步，读者应该熟练掌握这些操作。

- 第 5、6、7 章分别讲述了 MATLAB 中的基本数据类型，包括数值、结构体、元胞数组和字符串，第 8 章讲述了关系运算和逻辑运算，第 9 章讲述了程序控制流，这些内容是 MATLAB 编程语言的语法基础。

  理解各种数据类型、运算和程序控制流，是应用 MATLAB 语言进行编程，提高使用效率的前提。

- 第 10、11、12 章详细阐述了 MATLAB 中 M 文件编程的内容，主要包括 MATLAB 的函数，M 文件调试和管理等。

  这部分重点阐述了 MATLAB 中 M 文件编程的种类和基本流程，是通过实际的编程实践对语法基础部分内容的综合运用。

- 第 13 章介绍了 MATLAB 中另一种比较常用的数据结构——时间，以及此种数据结构在程序中的应用。

- 第 14~22 章重点讲述了在各种具体应用领域中的 MATLAB 实现，包括：矩阵代数、数据分析、数据插值、多项式、三次样条、傅里叶分析、最优化计算、微积分、常微分方程。

  这部分属于 MATLAB 的数值计算，也是 MATLAB 最精华的部分，这些章节简要回顾了数学理论，通过大量的应用实例讲述 MATLAB 应用。

- 第 23 章讲述二维绘图，第 24、25 章讲述三维绘图及其颜色、光影的修饰。

  这部分内容以 MATLAB 中的各种基本绘图指令和交互式环境为核心，目的在于使读者了解 MATLAB 中各种基本图形可视化方法的实现，并且初步介绍了 MATLAB 中图形的组织元素。

- 第 26 章“图像、声音和视频”，介绍了 MATLAB 中操作各种媒体文件的方法，对 MATLAB 中的图像相关操作及概念进行了深入的讲解。

- 第 27 章“图形的打印和导出”，讲述了 MATLAB 中保存绘图结果的各种方法，重点介绍了其中各种设置选项的意义和设置方法。

- 第 28 章“句柄图形对象”，讲述 MATLAB 中各种图形对象的组织形式，常用图形对象的基本属性和操作方法等。

  本章实际上是讲解MATLAB 内部各种图形组成元素的组织架构及其属性操作。这可以算是对前面几章介绍的直观图形结果的抽象总结。理解 MATLAB 中各种底层的句柄图形对象，是深入掌握 MATLAB 数据可视化技术的关键。任何高层的用法都是基于对这些底层句柄图形对象的操作实现的。

- 第 29 章介绍了 MATLAB 中图形用户界面（GUI）编程的内容。

  GUIDE 这一交互式图形界面为用户开发 MATLAB 的 GUI 程序提供了十分便利的开发环境。

- 第 30 章讲述了 MATLAB 类和面向对象编程，重点讲述了 MATLAB 中类的基本概念以及类的设计。

  MATLAB 的面向对象编程使得用户可以定义新的数据类型，并可方便地对其进行一系列的操作且不需了解其完成的细节，是扩展 MATLAB 功能的一个重要方面。

- 第 31 章讲述了 MATLAB 编程接口，强大的编程接口支持 MATLAB 与其他应用程序进行数据交换。

  MATLAB 编程接口大大方便了特定领域的用户，而且使得其他应用程序可以利用 MATLAB 中的强大功能。

- 第 32 章讲述了扩展 MATLAB 和 Java，MATLAB 与 Java 语言的接口。

  Java 是一种非常强大的语言，而 MATLAB 具有强大的计算功能，把这两种语言结合起来，能大大提高效率。

- 第 33 章讲述了 Windows 应用程序集成，它是 MATLAB 体系的一个重要功能。

  MATLAB 通过 COM、DDE、Notebook 等工具与其他的软件集成在一起，实现复杂的应用程序。

- 第 34 章讲述了 Simulink 交互式仿真集成环境，介绍了 Simulink 的基本模块、基本功能、以及如何使用 Simulink 进行仿真等内容。

  熟悉 Simulink 众多的功能强大模块，以及熟悉模块的基本操作是使用 Simulink 进行仿真和设计的基础。

- 第 35 章讲述了 MATLAB 高等数学计算实例，通过大量的高等数学应用实例介绍 MATLAB 函数的使用。

  熟练掌握 MATLAB 这一工具在数学计算的应用，可以从繁杂的计算中解脱出来，大大提高效率。

- 第 36 章讲述了 MATLAB 图形绘制实例，重点讲述了二维图形和三维图形的绘制。

  MATLAB 中丰富的图形函数，是实现数据可视化的重要组成部分，使用 MATLAB 图形绘制功能，用户能够方便直观地查看和分析个人数据。

- 第 37 章讲述了 MATLAB 扩展编程实例，列举了 MATLAB 与 VC++混合编程的几种主要方法。

## 光盘使用说明

本书附带光盘包含了全书所有实例对应的 MATLAB M 文件，所有代码按照章节存放在各个文件夹下，如 Ex-03 文件夹下存放第 3 章所有的实例代码，Ex-04 文件夹下存放第 4 章所有的实例代码，依此类推。在每一个文件夹下的 M 文件，其名称和书中的实例编号一一对应，如 Ex0301.m 文件对应于例 3-1，Ex0302.m 文件对应于例 3-2，依此类推。

读者可以通过运行光盘提供的代码文件，体会本书所有实例的效果。由于所有代码都是在 MATLAB 7（R14）下编写并调试通过，因此，使用本光盘中的实例前，读者需要安装 MATLAB 7（R14），并将包含待运行.m 文件的文件夹添加到 MATLAB 路径或设置为 MATLAB 当前目录。例如，运行 Ex0401.m，就要将包含此 M 文件的 Ex-04 文件夹添加到 MATLAB 路径，或者将其设置为 MATLAB 当前目录，然后通过命令窗口调用文件名；或者在 M-Editor 窗口打开代码文件并选择运行菜单的方式来运行此 M 文件。

## 本书读者对象

本书既可以作为 MATLAB 的参考手册，又适合作为本科教材和自学教材；既适合初级读者，又适合中高级读者；各章节之间既相互联系又相对独立，读者可根据自己的需要选择学习，结合光盘中的实例不断练习熟练掌握。

在本书编写过程中，得到了朱沭红策划编辑、孙学瑛编辑、许艳编辑的大力支持，在此对她们表示衷心的感谢！对 IBM 公司的肖静小姐、北京语言文化大学的小胖和北京大学的谢扬给予我们持续的鼓励和支持表示感谢，对资深软件工程师李彬先生的帮助表示感谢，对各位钻研 MATLAB 的网友给予的启发和帮助表示感谢。

由于时间仓促、作者水平和经验有限，书中错漏之处在所难免，敬请读者指正，我们的电子邮箱是：liu1984ming@sohu.com。

作者

2006 年 4 月

## 作者简介

王正林，北京科技大学博士 3 年级学生。精通 MATLAB 计算、编程和仿真。

刘明，生物科学专业本科毕业，现在清华大学攻读硕士学位。精通 MATLAB 计算、编程和图形图像，曾凭借对 MATLAB 的熟练应用在 CUMCM 和 MCM 竞赛中获得优异成绩。




Broadview®
WWW.BROADVIEW.COM.CN

Csdn

# 技术凝聚实力 专业创新出版

**博文视点**（www.broadview.com.cn）资讯有限公司是电子工业出版社、CSDN.NET、《程序员》杂志联合打造的专业出版平台。博文视点致力于——IT专业图书出版，为IT专业人士提供真正专业、经典的好书。

**请访问 www.dearbook.com.cn**（第二书店）购买优惠价格的博文视点经典图书。
**请访问 www.broadview.com.cn**（博文视点的服务平台）了解更多更全面的出版信息；您的投稿信息在这里将会得到迅速的反馈。

## 典藏外版精品




**JOLT大奖经典之作，关于交互系统设计的真知灼见！**

### 软件观念革命——交互设计精髓

[美]Alan Cooper，Robert Reimann 著
詹剑锋、张知非 等译 2005年6月出版
ISBN 7-121-01180-8 89.00元 650页

这是一本在交互设计前沿有着10年设计咨询经验及25年计算机工业界经验的卓越权威——VB之父ALAN COOPER撰写的设计数字化产品行为的启蒙书。



**全面阐释软件开发的最佳实践和重大原则！**

### 程序员修炼之道——从小工到专家

[美]Andrew Hunt，David Thomas 著
马维达 译
2004年4月出版 ISBN 7-5053-9719-2
48.00元 362页

本书由一系列独立的部分组成，涵盖的主题从个人责任、职业发展，直到用于使代码保持灵活、并且易于改编和复用的各种架构技术，利用许多富有娱乐性的奇闻轶事、有思想性的例子以及有趣的类比。




**设计心理学的经典之作！**
**中科院院士张钹亲自作序，人机交互专家叶展高度评价！**

### 情感化设计

[美]Donald A. Norman 著
付秋芳、程进三 译
2005年5月出版 ISBN 7-121-00940-4
36.00元 206页

设计的最高境界是什么？本书以独特细腻、轻松诙谐的笔法，以本能、行为和反思这三个设计的不同维度为基础，阐述了情感在设计中所处的重要地位与作用。




**软件管理方面的"MBA教程"的称号！荣获第15届JOLT大奖！**

### JOEL说软件

[美]Joel Spolsky 著
谭明金、王平 译
2005年9月出版 ISBN 7-121-01641-9
39.00元 301页

这是一本关于软件管理的随笔文集，这是一本会让你受益颇多的休闲之作。



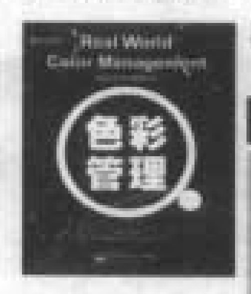

**被欧美许多重要大学用于"程序设计语言"或者"软件系统"课程！**

### 程序设计语言——实践之路

[美]Michael L.Scott 著
裘宗燕 译
2005年3月出版 ISBN 7-121-00900-5
88.00元 884页

这是一本很有特色的教材，其核心是讨论程序设计语言的工作原理和技术。

本书作者Michael Scott是计算机领域的著名学者，译者是北京大学的裘宗燕教授，他熟悉专业，译笔流畅，是一本难得的著、译双馨的佳作。




**北京印刷学院刘浩学教授翻译，方正色彩管理小组审校推荐！**

### 色彩管理

[美]Bruce Fraser，Chris Murphy，Fred Bunting 著
刘浩学、梁晓、武兵 等译
2005年7月出版 ISBN 7-121-01470-X
168.00元 504页

读懂它，不仅可以掌握精确一致的色彩复制技术，在最普及的图形图像软件中如何进行色彩管理，而且还可以知晓建立、评估和编辑ICC PROFILE；不仅可以知道色彩管理是怎么回事、如何做，而且知道为什么要这样做；不仅可以将色彩管理嵌入生产流程中，而且还能帮助改善生产流程，提高工作效率。

博文视点资讯有限公司
电　话：（010）51260888　传真：（010）51260888-802
E-mail：market@broadview.com.cn（市场）
editor@broadview.com.cn　jsj@phei.com.cn（投稿）
通信地址：北京市万寿路173信箱　北京博文视点资讯有限公司
邮　编：100036

电子工业出版社发行部
发 行 部：（010）88254055
门 市 部：（010）68279077　68211478
传　真：（010）88254050　88254060
通信地址：北京市万寿路173信箱
邮　编：100036

投　稿：editor@broadview.com.cn，jsj@phei.com.cn
销售咨询：market@broadview.com.cn
咨询电话：（010）51260888
Broadview 原创与引进并重，欢迎投稿

# 典藏本版精品




荣获 2004 年度“中国图书奖”和“全国优秀畅销书奖”！

## 编程高手箴言

梁肇新 编著
2003 年 11 月出版　ISBN 7-5053-9141-0
50.00 元（含光盘 1 张）　416 页

中国最具知名度的程序员之一，《超级解霸》作者梁肇新首部专著！

全书通篇没有时髦的 IT 新名词或新思想，而是踏踏实实地对很多知识进行了深刻的剖析，有助于为编程打下坚实的根基。

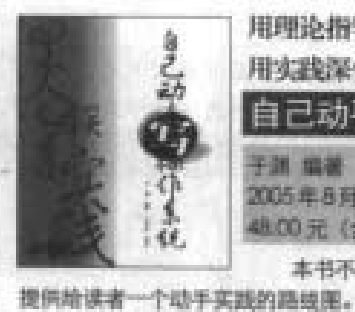



国内第一本重量级 Hibernate 图书。

## 深入浅出 Hibernate

夏昕、曹晓钢、唐勇 编著
2004 年 7 月出版　ISBN 7-121-00670-7
59.00 元　545 页

本书由互联网上影响广泛的开放文档 OpenDoc 系列自由文献首份文档“Hibernate 开发指南”发展而来。在编写过程中，进行了重新构思与组织，同时对内容的深度与广度进行了重点强化。



用理论指导动手实践
用实践深化理解理论

## 自己动手写操作系统

于渊 编著
2005 年 8 月出版　ISBN 7-121-01577-3
48.00 元（含光盘 1 张）　374 页

本书不同于其他的理论型书籍，而是提供给读者一个动手实践的路线图。

在详细分析操作系统原理的基础上，用丰富的实例代码，一步一步地指导读者用 C 语言和汇编语言编写出一个具备操作系统基本功能的操作系统框架。

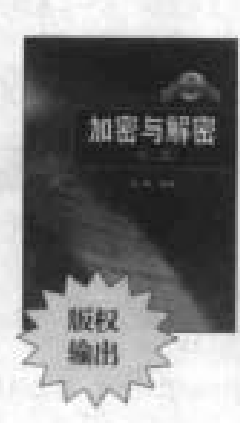



同类书销量第一！

## ERP 原理 设计实施（第 3 版）

罗鸿 编著
2005 年 4 月出版　ISBN 7-121-01059-3
38.00 元　384 页

本书对 ERP 相关知识的讨论涵盖了原理、设计与应用的全部过程。前两版出版后均引起了很大的社会反响，作者收到大量读者来信，并与读者进行了良好的交互。第 3 版再次增加了一些内容，更加贴近读者需要。




荣获 2003 年“全国优秀畅销书奖”，看雪论坛鼎立打造！

## 加密与解密（第二版）

段钢 编著
2003 年 6 月出版　ISBN 7-5053-8648-4
49.00 元（含光盘 1 张）　519 页

本书全面讲述了 Windows 平台下的最新软件加密与解密技术及相关解决方案，采用循序渐进的方式，从基本的跟踪调试到深层的拆解脱壳，从浅显的注册码分析到商用软件保护，几乎囊括了 Windows 下的软件保护的绝大多数内容。




本书通过多种典型实例详细介绍了在 Windows 系统下数据恢复技术的原理和方法。

## 数据恢复技术（第 2 版）

戴士剑、涂彦晖 编著
2005 年 3 月出版　ISBN 7-121-00756-8
69.00 元　711 页

本书内容包括：硬盘数据组织、文件系统原理、数据恢复技术、文档修复技术、密码遗失处理技术、数据安全技术和数据备份技术。作者戴士剑是国内知名数据恢复专家，有多年的数据恢复工作经验，为客户提供过上千次的数据恢复服务。




# 电子信息工程类图书



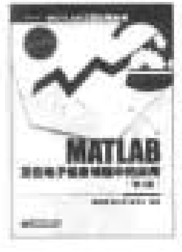
## 线性代数实践及 MATLAB 入门

陈怀琛、龚杰民 编著
2005 年 11 月出版
ISBN 7-121-01860-8
24.00 元
230 页



## 数字信号处理教程——MATLAB 释义与实现

陈怀琛 编著
2004 年 12 月出版
ISBN 7-121-00423-2
29.00 元
360 页



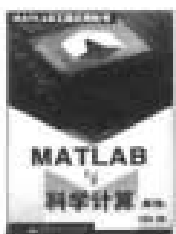
## MATLAB 及在电子信息课程中的应用（第 3 版）

陈怀琛 编著
2006 年 3 月出版
ISBN 7-121-02271-0
29.00 元
329 页



## MATLAB 教程

罗建军 主编　杨琦 副主编
2005 年 7 月出版
ISBN 7-121-01468-8
25.00 元
233 页



## MATLAB 与科学计算（第 2 版）

王沫然 编著
2003 年 9 月出版
ISBN 7-5053-9120-8
39.00 元
441 页



## 微型计算机控制技术实用教程

陈怀琛 编著
2004 年 12 月出版
ISBN 7-121-00423-2
29.00 元
360 页



## Keil Cx51 V7.0 单片机高级语言编程与 μVision2 应用实践

徐爱钧、彭秀华 编著
2004 年 6 月出版
ISBN 7-120-00057-8
69.00 元（含光盘 1 张）
706 页



## 可编程计算机控制器技术

齐蓉、肖维荣 编著
2005 年 11 月出版
ISBN 7-121-01813-6
35.00 元
400 页



# 《精通 MATLAB 7》读者调查表

尊敬的读者：

感谢您对我们的支持与爱护。为了今后为您提供更优秀的图书，请您抽出宝贵的时间将您的意见以下表的方式及时告知我们（可另附页）。我们将从中评选出热心读者若干名，免费赠阅我们以后出版的图书。

| 姓名： | 性别：□ 男 □ 女 | 年龄： | 职业： |
|---|---|---|---|
| 通信地址： | | | 邮政编码： |
| 电话： | 传真： | E-mail： | |

**1. 影响您购买本书的因素（可多选）：**

□ 封面封底　□ 价格　□ 内容提要、前言和目录　□ 书评广告　□ 出版物名声
□ 作者名声　□ 正文内容　□ 其他________

**2. 您对本书的满意度：**

从技术角度　□ 很满意　□ 比较满意　□ 一般　□ 较不满意　□ 不满意
□ 改进意见________

从文字角度　□ 很满意　□ 比较满意　□ 一般　□ 较不满意　□ 不满意
□ 改进意见________

从版面、封面设计角度　□ 很满意　□ 比较满意　□ 一般　□ 较不满意
□ 不满意　□ 改进意见________

**3. 您最喜欢书中的哪篇（或章、节）？请说明理由。**

**4. 您最不喜欢书中的哪篇（或章、节）？请说明理由。**

**5. 您希望本书在哪些方面进行改进？**

**6. 您感兴趣或希望增加的图书选题有：**

通信地址：北京万寿路 173 信箱　博文视点（100036）　电话：010-51260888
如果您对我们出版的图书有任何意见和建议，也可以发邮件给我们，我们将及时回复。
E-mail：jsj@phei.com.cn，editor@broadview.com.cn


www.broadview.com.cn

# 目　录

## 基 础 篇

# 实例目录




# 基 础 篇

- 基础入门
- MATLAB 桌面
- 数组及其操作
- 多维数组及其操作
- 数据类型概述和数值类型
- 结构体和元胞数组
- 字符串
- 关系运算和逻辑运算
- 程序控制流
- 函数
- M 文件调试和剖析
- 目录管理和文件 I/O
- MATLAB 中的时间
- 矩阵代数
- 数据分析
- 数据插值
- 多项式
- 三次样条
- 傅里叶分析
- 最优化计算
- 微积分
- 常微分方程
- 二维图形
- 三维图形
- 使用颜色和光影
- 图像、声音和视频
- 图形的打印和导出
- 句柄图形对象
- 图形用户界面（GUI）
- MATLAB 类和面向对象编程
- MATLAB 编程接口
- 扩展 MATLAB 和 Java
- Windows 应用程序集成
- Simulink 交互式仿真集成环境

# 第 1 章
# 基础入门

经过二十余年的补充与完善以及多个版本的升级换代，MATLAB 已发展至 7.x 版本，成为一个包含众多工程计算和仿真功能与工具的庞大系统，是目前世界上最流行的仿真计算软件之一。

## 1.1 MATLAB 发展历程

MATLAB 的产生是与数学计算紧密联系在一起的。1980 年，美国新墨西哥州大学计算机系主任 Cleve Moler 在给学生讲授线性代数课程时，发现学生在高级语言编程上花费很多时间，于是着手编写供学生使用的 Fortran 子程序库接口程序，取名为 MATLAB（即 Matrix Laboratory 的前三个字母的组合，意为"矩阵实验室"）。这个程序获得了很大的成功，受到学生的广泛欢迎。

20 世纪 80 年代初期，Moler 等一批数学家与软件专家组建了 MathWorks 软件开发公司，继续从事 MATLAB 的研究和开发，1984 年推出了第一个 MATLAB 商业版本，其核心用 C 语言编写。而后，MATLAB 又添加了丰富多彩的图形图像处理、多媒体、符号运算，以及与其他流行软件的接口功能，功能越来越强大。

MathWorks 公司正式推出 MATLAB 后，于 1992 年推出了具有划时代意义的 MATLAB 1.0 版本；1999 年推出的 MATLAB 5.3 版在很多方面进一步改进了 MATLAB 的功能，随之推出的全新版本 Simulink 3.0 也达到了很高的档次；2000 年 10 月推出的 MATLAB 6.0 版本，在操作界面上有了很大的改观，同时还给出了程序发布窗口、历史信息窗口和变量管理窗口等，为用户提供了极大的方便；2001 年 6 月，MATLAB 6.1 版即 Simulink 4.1 版问世，功能已经十分强大，其虚拟显示工具箱更给仿真结果在三维视景下显示带来了新的解



决方案；2003 年 6 月推出了 MATLAB Release 13，即 MATLAB 6.5/Simulink 5.0，在核心数值算法、界面设计、外部接口和应用桌面等诸多方面有了极大的改进；2004 年 9 月正式推出 MATLAB Release 14，即 MATLAB 7.0/Simulink 6.0，其功能在原有的基础上又有了进一步的改进，是 MATLAB 目前最新的版本。

MATLAB 经过二十多年的研究与不断完善，现已成为国际上最流行的科学计算与工程计算软件工具之一，现在的 MATLAB 已经不仅仅是最初的"矩阵实验室"了，它已发展成为一种具有广泛应用前景的、全新的计算机高级编程语言，可以说它是"第四代"计算机语言。自 20 世纪 90 年代以来，美国和欧洲大学将 MATLAB 正式列入研究生和本科生的教学计划，MATLAB 软件已成为应用代数、自动控制理论、数理统计、数字信号处理、时间序列分析和动态系统仿真等课程的基本教学工具，成为学生所必须掌握的基本软件之一。在设计研究单位和工业界，MATLAB 也成为工程师们必须掌握的一种工具，被认为是进行高效研究与开发的首选软件工具。

## 1.2 MATLAB 系统结构

MATLAB 系统由 MATLAB 开发环境、MATLAB 数学函数库、MATLAB 语言、MATLAB 图形处理系统和 MATLAB 应用程序接口（API）五大部分构成。

### 1. MATLAB 开发环境

MATLAB 开发环境是一套方便用户使用的 MATLAB 函数和文件工具集，其中许多工具是图形化用户接口。它是一个集成的工作空间，允许用户输入输出数据，并提供了 M 文件的集成编译和调试环境，包括 MATLAB 桌面、命令窗口、M 文件编辑调试器、MATLAB 工作空间和在线帮助文档。

### 2. MATLAB 数学函数库

MATLAB 数学函数库包括了大量的计算算法，从基本运算，如加法、正弦等，到复杂算法，如矩阵求逆、贝塞尔函数和快速傅里叶变换等。

### 3. MATLAB 语言

MATLAB 语言是一种高级的基于矩阵/数组的语言，它有程序流控制、函数、数据结构、输入/输出和面向对象编程等特色。用户既可以用它来快速编写简单的程序，也可以编写庞大复杂的应用程序。

### 4. MATLAB 图形处理系统

图形处理系统使得 MATLAB 能方便地图形化显示向量和矩阵，而且能对图形添加标注和打印。它包括强大的二维、三维图形函数、图像处理和动画显示等函数。



#### 5. MATLAB 应用程序接口（API）

MATLAB 应用程序接口（API）是一个使 MATLAB 语言能与 C，Fortran 等其他高级编程语言进行交互的函数库，该函数库的函数通过调用动态链接库（DLL）实现与 MATLAB 文件的数据交换，其主要功能包括在 MATLAB 中调用 C 和 Fortran 程序，以及在 MATLAB 与其他应用程序间建立客户/服务器关系。

## 1.3 MATLAB 7 工具箱

MATLAB 拥有一个专用的家族产品，用于解决不同领域的问题，称之为工具箱（Toolbox）。工具箱用于 MATLAB 的计算和画图，通常是 M 文件和高级 MATLAB 语言集合，使用户可以方便地修改函数和源代码，或增加新的函数。用户还可以结合不同的工具箱中的技术来设计针对某个问题的解决方案。MATLAB 每年都会增加一些新的工具箱，因此，在一般情况下，工具箱的列表不是固定不变的。有关 MATLAB 工具箱的最新信息可以在 http://www.mathworks.com/products 中看到。

较为常见的 MATLAB 工具箱包括：

#### 1. 控制类工具箱

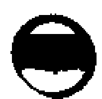
（1）控制系统工具箱（Control Systems Toolbox）
（2）系统辨识工具箱（System Identification Toolbox）
（3）鲁棒控制工具箱（Robust Control Toolbox）
（4）模糊逻辑工具箱（Fuzzy Logic Toolbox）
（5）神经网络工具箱（Neural Network Toolbox）
（6）频域系统辨识工具箱（Frequency Domain System Identification Toolbox）
（7）模型预测控制工具箱（Model Predictive Control Toolbox）
（8）多变量频率设计工具箱（Multivariable Frequency Design Toolbox）

#### 2. 应用数学类工具箱

（1）最优工具箱（Optimization Toolbox）
（2）样条工具箱（Spline Toolbox）
（3）统计工具箱（Statistics Toolbox）
（4）偏微分方程工具箱（Partial Differential Equation Toolbox）

#### 3. 信号处理类工具箱

（1）信号处理工具箱（Signal Processing Toolbox）
（2）滤波器设计工具箱（Filter Design Toolbox）
（3）通信工具箱（Communication Toolbox）
（4）小波分析工具箱（Wavelet Toolbox）





（5）高阶谱分析工具箱（Higher Order Spectral Analysis Toolbox）

**4．其他常用的工具箱**

（1）符号数学工具箱（Symbolic Math Toolbox）
（2）虚拟现实工具箱（Virtual Reality Toolbox）
MATLAB 7 对以前的版本进行了升级，新增了以下一些功能强大的工具箱：
（1）定点工具箱（Fixed-Point Toolbox）
（2）遗传算法和直接搜寻工具箱（Genetic Algorithm and Direct Search Toolbox）
（3）射频工具箱（RF Toolbox）
（4）生物信息工具箱（Bioinformatics Toolbox）
（5）OPC 工具箱（OPC Toolbox）

## 1.4 MATLAB 7/Simulink 6 最新特点

MATLAB 既是高级科学计算语言，又是进行数据分析和算法开发的集成开发环境。MATLAB 7.0 在 MATLAB 6.5 的基础上进行了很多改善，Simulink 6.0 在 Simulink 5.0 的基础上也增加了不少新功能，使得这一软件的性能大大提高。

### 1.4.1 MATLAB 7 最新特点

MATLAB 7.0 在编程环境、代码效率、数据可视化、数学计算和文件 I/O 等方面进行了升级，其中包括：

**1．开发环境方面**

（1）重新设计的桌面环境，针对多文档界面应用提供了简便的管理和访问方法，允许用户自定义桌面外观，创建常用命令的快捷方式；

（2）增强数组编辑器（Array Editor）和工作空间浏览器（Workspace Browser）功能，用于数据的显示、编辑和处理；

（3）在当前目录浏览器（Current Directory Browser）工具中，增加代码效率分析和覆盖度分析等功能；

（4）M-Lint 编码分析辅助用户完成程序性能分析，提高程序执行效率；

（5）增强 M 文件编辑器（M-Editor），支持多种格式源代码文件可视化编辑，如 C/C++，HTML，Java 等。

**2．编程方面**

（1）支持创建嵌套函数（Nested Function），提供更灵活的代码模块化方式；

（2）匿名函数（Anonymous Function）功能，支持在命令行或者脚本文件中创建单行函数（Single Line Function）；



（3）支持条件分支断点，可以在条件分支语句中进行程序中断调试；

（4）模块化注释，支持为代码段注释。

**3. 数学运算方面**

（1）支持整数算术运算；

（2）支持单精度数据类型运算，包括基本算术运算、线性代数和快速傅里叶变换（FFT）等；

（3）使用更强大的计算算法包 Qhull 2002.1，提供更丰富的算法支持；

（4）insolve 函数用于线性代数方程求解；

（5）ODE 求解器能够处理隐性微分方程组以及多点边界问题。

**4. 图形和 3D 可视化方面**

（1）新图形窗体界面；

（2）直接从图形窗体生成 M 代码，可以完成用户自定义绘图；

（3）增强图形窗体注释；

（4）数据浏览工具（Data Exploration Tools）提供丰富的数据观测手段；

（5）自定义图形对象，提供丰富的图形显示能力；

（6）GUIDE 新增对用户界面面板和 ActiveX 控件支持；

（7）增强句柄图形对象，支持完整的 TeX 和 LaTeX 字符集。

**5. 文件 I/O 和外部接口方面**

（1）新增文件 I/O 函数，支持任意格式文本数据文件读取，并且支持写入 Excel 和 HDF5 格式数据文件；

（2）具有压缩功能的 MAT 文件格式，支持快速数据文件 I/O 能力；

（3）javaaddpath 函数，无需重新启动 MATLAB 即可完成 Java 类的加载、删除等功能；

（4）支持 COM、服务器事件以及 VBS；

（5）支持 SOAP，使用网络服务；

（6）支持 FTP 对象，可以直接访问 FTP 服务器；

（7）支持 Unicode 编码格式，增强 MAT 文件字符集。

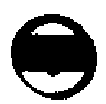
**6. 性能与系统平台支持方面**

（1）JIT 加速器支持所有数值数据类型；

（2）Windows XP 系统下支持 3GB 内存访问。

## 1.4.2 Simulink 6 最新特点

Simulink 是一个交互式动态系统建模、仿真和分析图形环境，可用于基于模型的嵌入式系统开发。Simulink 可以针对控制系统、信号处理和通信系统等进行系统建模、仿真和分析等工作。Simulink 6 改善了性能，并且针对大规模系统开发进行了性能优化，主要包





括以下方面：

### 1. 大系统建模方面

（1）支持将大系统模型分割为不同的文件，每个文件为独立的系统模型；

（2）支持系统不同模型文件独立仿真测试；

（3）增强系统集成手段，支持配置管理和版本控制软件；

（4）递增式模型加载与代码生成功能；

（5）针对大模型系统提高运行速度和效率；

（6）模型工作空间（Model Workspace），每个模型都用于独立的工作空间用于参数管理和数据处理；

（7）增强总线支持。

### 2. Simulink 和 Stateflow 方面

（1）统一的模型浏览器（Model Explorer），用于浏览、维护、配置、搜索、定义所有模型中相关的参数、数据对象和属性；

（2）统一的仿真和代码生成选项；

（3）支持创建并保存多种仿真和代码生成选项；

（4）数据管理和可视化；

（5）新增数据对象属性；

（6）可选数据记录增加测试点，无需在模型中增加额外的模块；

（7）I/O 管理，可以将必要的信号源和信宿连接到模型而不需要增加模块。

### 3. MATLAB 支持方面

使用嵌入式 MATLAB（Embedded MATLAB）功能引入算法，并支持 C 代码生成。

## 1.5 MATLAB 启动和退出

以 Windows 操作系统为例，进入 Windows 后，选择“开始”→“程序”→“MATLAB 7.0”，便可以进入如图 1-1 所示的 MATLAB 主窗口。如果安装时选择在桌面上生成快捷方式，也可以点击快捷方式直接启动。

在启动 MATLAB，命令编辑区显示帮助信息后，将显示符号“|”，表示 MATLAB 已准备好，正等待用户输入命令，这时，在提示符“|”后面键入命令，按下回车键，MATLAB 就会解释执行所输入的命令，并在命令后面给出计算结果。如果在输入命令后以分号结束，则不会显示结果。

退出 MATLAB 系统的方式有两种：

（1）在文件菜单（File）中选择“Exit”或“Quit”；

（2）用鼠标单击窗口右上角的关闭图标“X”。

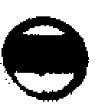





图 1-1　MATLAB 主窗口

# 1.6　MATLAB 基本特色

## 1.6.1　常量与变量

### 1．常量

常量是指那些在 MATLAB 中已预先定义其数值的变量，默认的常量如表 1-1 所示。

表 1-1　MATLAB 默认常量

| 名　称 | 说　明 |
|---|---|
| pi | 圆周率 |
| INF（或 inf） | 无穷大 |
| NaN（或 nan） | 代表不定值（即 0/0） |
| realmax | 最大的正实数 |
| realmin | 最小的正实数 |
| eps | 浮点数的相对误差 |
| i（或 j） | 虚数单位，定义为 $\sqrt{-1}$ |
| nargin | 函数实际输入参数个数 |
| nargout | 函数实际输出参数个数 |
| ANS（或 ans） | 默认变量名，以应答最近一次操作运算结果 |

### 2．变量

变量是数值计算的基本单元。与 C 语言等其他高级语言不同，MATLAB 语言中的变量



无须事先定义，一个变量以其名称在语句命令中第一次合法出现而定义，运算表达式变量中不允许有未定义的变量；MATLAB 也不需要预先定义变量的类型，它会自动生成变量，并根据变量的操作确定其类型。

（1）MATLAB 变量命名规则

MATLAB 中的变量命名规则如下：

① 变量名区分大小写，因此 $A$ 与 $a$ 表示的是不同的变量；

② 变量名以英文字母开始，第一个字母后可以使用字母、数字和下划线，但不能使用空格和标点符号；

③ 变量名长度不得超过 31 位，超过的部分将被忽略；

④ 某些常量也可以作为变量使用，如 i 在 MATLAB 中表示虚数单位，但也可以作为变量使用。

（2）MATLAB 变量的显示

任何 MATLAB 语句的执行结果都可以在屏幕上显示，同时赋值给指定的变量，没有指定变量时，MATLAB 将结果赋值给一个特殊的变量 ans。数据的显示格式由 format 命令控制。format 只是影响结果的显示，不影响其计算与存储。MATLAB 总是以双字长浮点数（双精度）来执行所有的运算。如果结果为整数，则显示没有小数；如果不是整数，则输出形式可为表 1-2 所示的几种形式。

**表 1-2　MATLAB 的数据显示格式**

| 格　式 | 含　义 |
|---|---|
| format (short) | 短格式（5 位定点数） |
| format long | 长格式（15 位定点数） |
| format short e | 短格式 e 方式 |
| format long e | 长格式 e 方式 |
| format bank | 2 位十进制 |
| format hex | 十六进制格式 |

（3）MATLAB 变量的存储

① 工作空间中的变量可以用 save 命令存储到磁盘文件中。

键入命令 save　<文件名>，将工作空间中全部变量存到<文件名>.mat 文件中去，若省略<文件名>选项则存入文件 matlab.mat 中；命令 save　<文件名>　<变量名集>将<变量名集>指出的变量存入文件<文件名>.mat 中。

② 用命令 load 可将变量从磁盘文件读入 MATLAB 的工作空间。

键入命令 load　<文件名>，将<文件名>指出的磁盘文件中的数据依次读入名称与<文件名>相同的工作空间中的变量中去。

若省略<文件名>，则从 matlab.mat 中读入所有数据。

③ 用 clear 命令可清除工作空间中现存的变量。

**例 1-1**　数据的存取。

**解**：（1）建立用户目录，并使之成为当前目录，保存数据



```
x=[1,2]   % 输入数据
mkdir('c:\','my_dir');          %在 C 盘上创建目录 my_dir
cd c:\my_dir                    %使 c:\my_dir 成为当前目录
save  saf  x                    %选择内存中的 x 变量保存为 saf.mat 文件
dir                             %显示目录上的文件
```

输出为：

```
>> In mkdir at 116
.          ..          saf.mat
```

（2）清空内存，从 saf.mat 向内存装载变量 $x$

```
>> clear                        %清除内存中的全部变量
load   saf   x                  %把 saf.mat 文件中的 x 变量装入内存
>> who                          %检查内存中有什么变量
Your variables are:
x
```

需要注意的是：如果一组数据是经过长时间的复杂计算后获得的，为避免再次重复计算，常使用 save 加以保存。此后，每当需要，都可通过 load 重新获取这组数据。这种处理模式常在实际中被采用。

### 1.6.2 MATLAB 基本运算

在 MATLAB 下进行基本数学运算，只需将运算式直接输入提示符（>>）之后，并按下 Enter 键即可。例如：

```
>> (5*2+1.3+1.8)*10/25
ans =
    5.2400
```

MATLAB 会将运算结果直接存入一变量 ans，代表 MATLAB 运算后的答案（answer），并显示其数值于屏幕上。

也可将上述运算式的结果设定给另一个变量 $x$。

```
>> x=(5*2+1.3+1.8)*10/25
x =
    5.2400
```

此时 MATLAB 会直接显示 $x$ 的值。

MATLAB 能识别所有一般常用的加（+）、减（–）、乘（*）、除（/）算术运算符号，以及幂次运算（^）。

需要注意的是：MATLAB 将所有变量均存成 double 的形式，所以不需经过变量定义，MATLAB 会自动进行内存的使用和清除，而不必像 C 语言由使用者一一指定。这些功能使得 MATLAB 易学易用，使用者可专心致力于编写程序，而不会被软件枝节问题所干扰。

如果不想让 MATLAB 每次都显示运算结果，只需在运算式最后加上分号（;）即可，如下例。



```
>> x=(5*2+1.3+1.8)*10/25;
```

若要显示变量 $x$ 的值，直接键入 $x$ 即可：

```
>> x
    x =
        5.2400
```

## 1.6.3 MATLAB 基本函数

### 1. 基本数学函数

MATLAB 具有强大的计算功能，它提供了大量的函数，方便使用者进行计算。常用的基本数学函数如表 1-3 所示。

表 1-3 常用的基本数学函数

| 函数名 | 含义 |
|---|---|
| abs(x) | 数的绝对值或向量的长度 |
| sqrt(x) | 开平方 |
| angle(z) | 复数 z 的相角 |
| real(z) | 复数 z 的实部 |
| imag(z) | 复数 z 的虚部 |
| conj(z) | 复数 z 的共轭复数 |
| round(x) | 四舍五入至最近整数 |
| fix(x) | 无论正负，舍去小数至最近整数 |
| floor(x) | 地板函数，即舍去正小数至最近整数 |
| ceil(x) | 天花板函数，即加入正小数至最近整数 |
| rat(x) | 将实数 x 化为分数表示 |
| rats(x) | 将实数 x 化为多项分数展开 |
| rem(x,y) | 求 x 除以 y 的余数 |
| gcd(x,y) | 整数 x 和 y 的最大公因数 |
| lcm(x,y) | 整数 x 和 y 的最小公倍数 |
| log10(x) | 以 10 为底的对数 |
| log2(x) | 以 2 为底的对数 |
| log(x) | 以 e 为底的对数，即自然对数 |
| pow2(x) | 2 的指数 |
| exp(x) | 自然指数 |
| sign(x) | 符号函数 |

### 2. 基本三角函数

MATLAB 常用的基本三角函数如表 1-4 所示。

表 1-4　常用的基本三角函数

| 函数名 | 含　义 |
| --- | --- |
| sin(x) | 正弦函数 |
| cos(x) | 余弦函数 |
| tan(x) | 正切函数 |
| asin(x) | 反正弦函数 |
| acos(x) | 反余弦函数 |
| atan(x) | 反正切函数 |
| atan2(x,y) | 四象限的反正切函数 |
| sinh(x) | 超越正弦函数 |
| cosh(x) | 超越余弦函数 |
| tanh(x) | 超越正切函数 |
| asinh(x) | 反超越正弦函数 |
| acosh(x) | 反超越余弦函数 |
| atanh(x) | 反超越正切函数 |

## 1.6.4　向量

### 1．向量基本操作

向量是 MATLAB 的基本运算单元，下面通过例子介绍向量的基本操作。

```
>> x=[1,2,3,4,5]                 %以行向量（数组）方式给 x 赋值
   t=[1;2;3;4;5]                 %以列向量（数组）方式给 t 赋值
   y=(x(3)+x(5))/2*x(4)          %x(3)调用 x 中的第 3 个元素
   z=sqrt(x)                     %每个元素开方
   u=x*t                         %向量的内积（u 为向量 x 的模的平方）
```

输出为：

```
x =
    1     2     3     4     5
t =
    1
    2
    3
    4
    5
y =
   16
z =
   1.0000    1.4142    1.7321    2.0000    2.2361
u =
   55
```

也可以随意更改、增加或删除向量中的元素，如下例所示：

```
>> x=[1,2,3,4,5]
   x(2)=20    % 更改第 2 个元素
```





输出为：

```
x =
     1    20     3     4     5
x(6)=16   % 加入第6个元素
```

输出为：

```
x =
     1    20     3     4     5    16
x(2)=[]   % 删除第2个元素
```

输出为：

```
x =
     1     3     4     5    16
```

2. 向量的操作函数

MATLAB 提供了常用的向量运算函数，如表 1-5 所示。

表 1-5 常用的向量运算函数

| 函数名 | 含义 |
|---|---|
| min(*x*) | 向量 *x* 的元素的最小值 |
| max(*x*) | 向量 *x* 的元素的最大值 |
| mean(*x*) | 向量 *x* 的元素的平均值 |
| median(*x*) | 向量 *x* 的元素的中位数 |
| std(*x*) | 向量 *x* 的元素的标准差 |
| diff(*x*) | 向量 *x* 的相邻元素的差 |
| sort(*x*) | 对向量 *x* 的元素进行排序 |
| length(*x*) | 向量 *x* 的元素个数 |
| norm(*x*) | 向量 *x* 的欧氏（Euclidean）长度 |
| sum(*x*) | 向量 *x* 的元素总和 |
| prod(*x*) | 向量 *x* 的元素总乘积 |
| cumsum(*x*) | 向量 *x* 的累计元素总和 |
| cumprod(*x*) | 向量 *x* 的累计元素总乘积 |
| dot(*x*, *y*) | 向量 *x* 和 *y* 的内积 |
| cross(*x*, *y*) | 向量 *x* 和 *y* 的外积 |

## 1.7 小结

本章介绍 MATLAB 的产生与发展过程，对该软件最新版本 MATLAB 7 的常用工具箱及最新特点进行了介绍。通过阅读本章，读者对 MATLAB 的发展历程及基本特点能有一个比较全面的了解。

# 第 2 章

# MATLAB 桌面

MATLAB 7 为用户提供了全新的桌面操作环境，了解并熟悉这些桌面操作环境是使用 MATLAB 的基础。下面介绍 MATLAB 的启动、主要功能菜单、命令窗口、工作空间、文件管理和帮助管理等。

## 2.1 MATLAB 主菜单及功能

打开 MATLAB 主窗口后，即弹出其主菜单栏，主菜单栏各菜单项及其下拉菜单的功能如下所述。

### 1．File 主菜单项

单击 File 主菜单项或同时按下“Alt+F”组合键，弹出如图 2-1 所示的 File 下拉菜单。其中，带下划线的字母表示快捷键，即按下该字母键也可执行相同的功能。

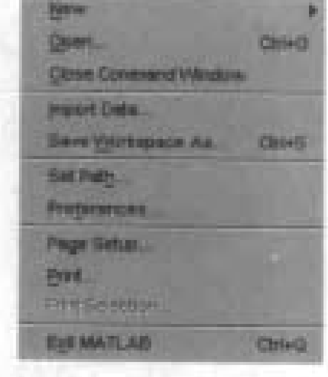


图 2-1 File 下拉菜单

（1）New：用于建立新的.m 文件、图形、模型和图形用户界面；

（2）Open：用于打开 MATLAB 的.m 文件、.fig 文件、.mat 文件、.mdl 文件、.cdr 文件等，也可通过快捷键“Ctrl+O”来实现此项操作；

（3）Close Command Window：关闭命令窗口；

（4）Import Data：用于从其他文件导入数据，单击该选项后弹出对话框，提示用户选择导入文件的路径和位置；

（5）Save Workspace As：用于把工作空间的数据存放到相应路径的文件中；

（6）Set Path：设置工作路径；





（7）Preferences：用于设置命令窗的属性，单击该选项，弹出如图2-2所示属性画面：

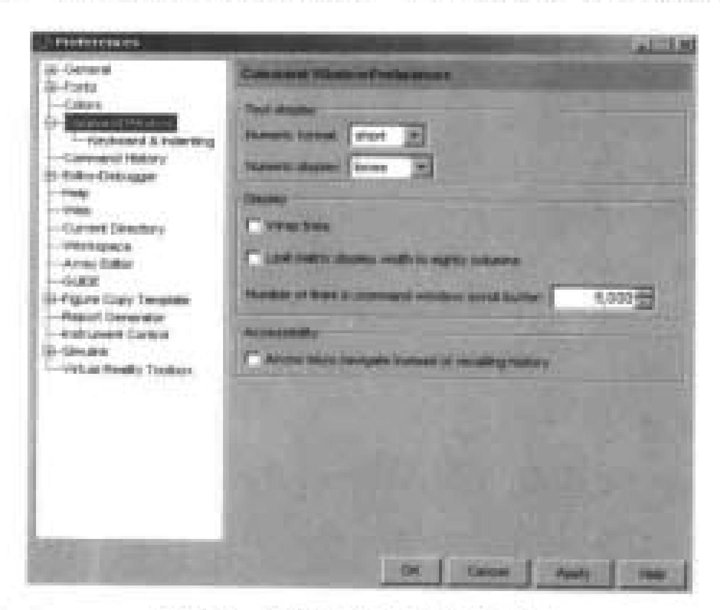


图2-2 设置命令窗口属性对话框

（8）Page Setup：用于页面设置；

（9）Print：用于设置打印属性；

（10）Print Selection：用于对选择的文件数据进行打印设置；

（11）Exit MATLAB：退出MATLAB桌面操作环境。

### 2. Edit 主菜单项

点击Edit主菜单项或同时按下"Alt+E"组合键，弹出如图2-3所示的下拉菜单。

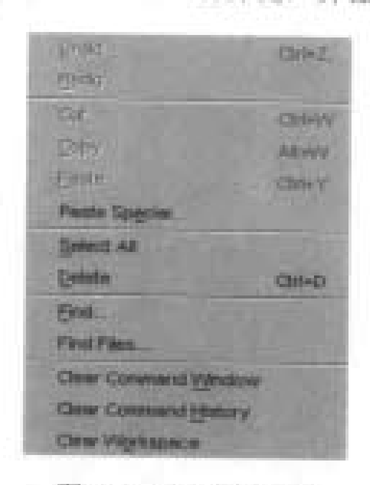




图2-3 Edit 下拉菜单

（1）Undo：用于撤销上一步操作，也可通过快捷键"Ctrl+Z"来实现；

（2）Redo：用于重新执行上一步操作；

（3）Cut：用于剪切选中的对象，也可通过快捷键"Ctrl+W"来实现；

（4）Copy：用于复制选中的对象，也可通过快捷键"Alt+W"来实现；

（5）Paste：用于粘贴剪贴板上的内容，也可通过快捷键"Ctrl+Y"来实现；
（6）Paste Special：用于特定内容的粘贴；
（7）Select All：用于全部选择；
（8）Delete：用于删除所选的对象，也可通过快捷键"Ctrl+D"来完成；
（9）Find：用于查找所需选择的对象；
（10）Find Files：用于查找所需文件；
（11）Clear Command Window：用于清除命令窗口区的对象；
（12）Clear Command History：用于清除命令窗口区的历史记录；
（13）Clear Workspace：用于清除工作区的对象。

### 3. Debug 主菜单项

单击 Debug 主菜单项或同时按下"Alt+B"组合键，弹出如图 2-4 所示的下拉菜单。

（1）Open M-Files when Debugging：用于调试时打开 M 文件；

（2）Step：用于单步调试程序，也可通过快捷键"F10"来实现；

（3）Step in：用于单步调试进入子函数，也可通过快捷键"F11"来实现；

（4）Step Out：用于单步调试从子函数跳出，也可通过快捷键"Shift+F11"来实现；

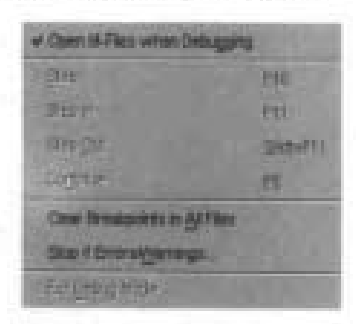


图 2-4 Debug 下拉菜单

（5）Continue：程序执行到下一断点，也可通过快捷键"F5"来实现；
（6）Clear Breakpoints in All Files：清除所有打开文件中的断点；
（7）Stop if Errors/Warnings：在程序出错或报警处停止往下执行；
（8）Exit Debug Mode：退出调试模式。

### 4. Desktop 主菜单项

单击 Desktop 主菜单项或同时按下"Alt+D"键，弹出如图 2-5 所示的下拉菜单。

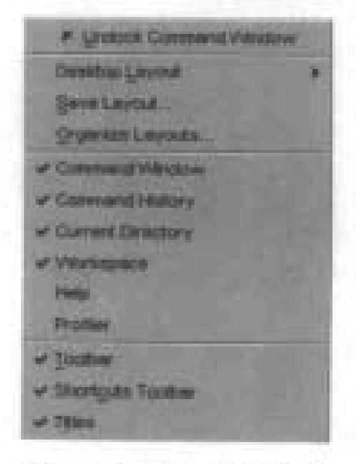


图 2-5 Desktop 下拉菜单



（1）Undock Command Window：将命令窗口变为全屏显示，并设为当前活动窗口；

（2）Desktop Layout：单击该项后，弹出如图2-6所示的子菜单，用于工作区的设置。其设置选项包括系统默认设置项（Default）、单独命令窗口项（Command Window Only）、命令历史窗口和命令窗口项（History and Command Window）、全部标签项显示（All Tabbed）；

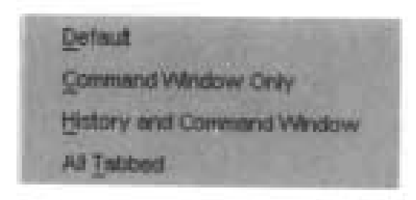



图2-6 Desktop Layout弹出子菜单

（3）Save Layout：保存选定的工作区设置；

（4）Organize Layouts：管理保存的工作区设置；

（5）Command Window：命令窗口项，选择该项，屏幕上便会显示该窗口；

（6）Command History：命令历史窗口项，选择该项，屏幕上便会显示该窗口；

（7）Current Directory：当前路径窗口项，选择该项，屏幕上便会显示该窗口；

（8）Workspace：工作窗口项，选择该项，屏幕上便会显示该窗口；

（9）Help：帮助窗口项，选择该项，屏幕上便会显示该窗口；

（10）Profiler：轮廓图窗口项，选择该项，屏幕上便会显示该窗口；

（11）Toolbar：显示或隐藏工具栏选项；

（12）Shortcuts Toolbar：显示或隐藏快捷方式选项；

（13）Titles：显示或隐藏标题栏选项。

### 5．Window主菜单项

单击Window主菜单项或同时按下“Alt+W”键，弹出如图2-7所示的下拉菜单。

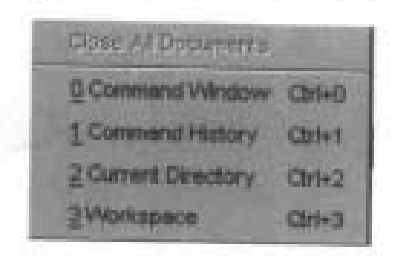



图2-7 Window下拉菜单

（1）Close All Documents：关闭所有文档；

（2）0 Command Window：选定命令窗口为当前活动窗口，也可通过快捷键“Ctrl+0”来实现；

（3）1 Command History：选定命令历史窗口为当前活动窗口，也可通过快捷键“Ctrl+1”来实现；

（4）2 Current Directory：选定当前路径窗口为当前活动窗口，也可通过快捷键“Ctrl+2”来实现；

（5）3 Workspace：选定工作空间窗口为当前活动窗口，也可通过快捷键“Ctrl+3”来实现。

### 6. Help 主菜单项

单击 Help 主菜单项或同时按下“Alt+H”组合键，弹出如图 2-8 所示的下拉菜单。

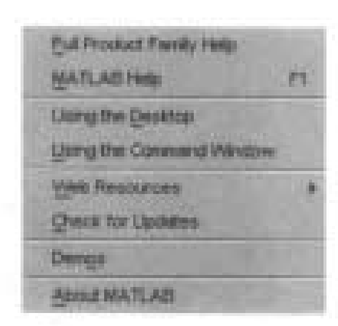



图 2-8 Help 下拉菜单

（1）Full Product Family Help：显示所有 MATLAB 产品的帮助信息；

（2）MATLAB Help：启动 MATLAB 帮助；

（3）Using the Desktop：启动 Desktop 的帮助；

（4）Using the Command Window：启动命令窗口的帮助；

（5）Web Resources：显示因特网上一些相关的资源网址；

（6）Check for Updates：检查软件是否更新；

（7）Demos：调用 MATLAB 所提供的范例程序；

（8）About MATLAB：显示有关 MATLAB 的信息。

## 2.2 MATLAB 命令窗口

MATLAB 的命令窗口如图 2-9 所示，它用于 MATLAB 命令的交互操作，具有两大主要功能：

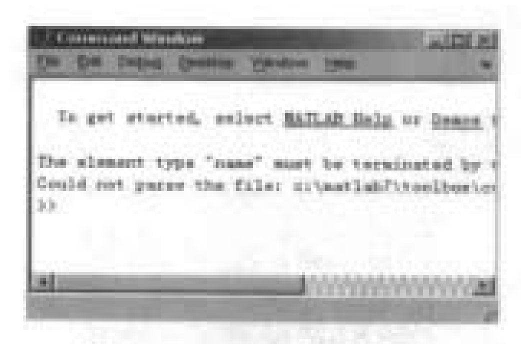


图 2-9 MATLAB 的命令窗口

（1）提供命令输入的操作平台，用户通过该窗口输入命令和数据；

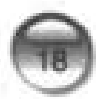
（2）提供命令执行结果的显示平台，该窗口显示命令执行的结果。

在计算机上安装 MATLAB 之后，双击 MATLAB 图标，就可以进入命令窗口，此时意味着系统处于准备接受命令的状态，可以在命令窗口中直接输入命令语句。

MATLAB 语句形式如下所示：

>>变量=表达式；

它通过等号将表达式的值赋予变量。当键入回车键后，该语句被执行。语句执行完毕，






窗口自动显示出语句执行的结果。如果不希望结果被显示，只要在语句之后加上一个分号（;）即可。此时尽管结果没有显示，但它依然被赋值，并且 MATLAB 在工作空间中为之分配了内存。

使用方向键和控制键可以编辑、修改已输入的命令，向上方向键“↑”用于回调上一行命令，向下方向键“↓”用于回调下一行命令。使用命令字 more off 表示不允许分页，more on 表示允许分页，more(n)表示指定每页输出的行数。键入回车键前进一行，空格键显示下一页，键入“q”命令字结束当前显示。

如果命令语句超过一行或者太长，希望分行输入，则可以使用多行命令继续输入。如输入下列式子时，可以通过两行输入：

```
S=1-12+13+4+…
   9-4-18;
```

## 2.3　MATLAB 工作空间

MATLAB 的工作空间（Workspace），如图 2-10 所示。

图 2-10　MATLAB 的工作空间

工作空间包含了一组可以在命令窗口中调整（使用）的参数，常用的工作空间操作的命令及功能如表 2-1 所示。

表 2-1　工作空间常用操作命令及功能

| 命　令 | 功　能 |
|---|---|
| who | 显示当前工作空间中所有变量的一个简单列表 |
| whos | 列出变量的大小、数据格式等详细信息 |
| clear | 清除工作空间中所有的变量 |
| clear　变量名 | 清除指定的变量 |
| load | 从磁盘文件中恢复变量 |
| save | 保存工作空间变量 |
| pack | 整理工作空间内存 |
| size(变量名) | 显示当前工作空间中变量的尺寸 |
| length(变量名) | 显示当前工作空间中变量的长度 |
| disp(变量名) | 显示当前工作空间中变量 |



MATLAB 提供了以下保存和载入 workspace 内变量的命令。

（1）save <filename> <variables>

将变量列表 variables 所列出的变量保存到磁盘文件 filename 中，variables 所表示的变量列表中不能用逗号，各个不同的变量之间只能用空格来分隔。未列出 variables 时，表示将当前工作空间中所有变量都保持到磁盘文件中。默认的磁盘文件扩展名为“.mat”，可以使用“-”定义不同的存储格式（ASCII、V4 等）。

（2）load <filename> <variables>

将以前用 save 命令保存的变量 variables 从磁盘文件中调入 MATLAB 工作空间。用 load 命令调入的变量，其名称为用 save 命令保存时的名称，取值也一样。variables 所表示的变量列表中，不能用逗号，各个不同的变量之间只能用空格来分隔。未列出 variables 时，表示将磁盘文件中的所有变量都调入工作空间。

（3）quit 或 exit

quit 或 exit 命令，退出工作空间。

## 2.4 MATLAB 文件管理

MATLAB 提供了一组文件管理命令，包括列文件名、显示或删除文件、显示或改变当前目录等，相关的命令及功能如表 2-2 所示。

表 2-2 MATLAB 常用文件管理命令

| 命　令 | 功　能 |
|---|---|
| what | 显示当前目录下所有与 MATLAB 相关的文件及它们的路径 |
| dir | 显示当前目录下所有的文件 |
| which | 显示某个文件的路径 |
| cd path | 由当前目录进入 path 目录 |
| type filename | 在命令窗口中显示文件 filename |
| delete filename | 删除文件 filename |
| cd .. | 返回上一级目录 |
| cd | 显示当前目录 |

## 2.5 MATLAB 帮助使用

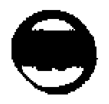
MATLBA 的所有函数都是以逻辑群组方式进行组织的，而 MATLAB 的目录结构就是以这些群组方式来编排的，几个常用的帮助命令如下：

（1）helpwin 帮助窗口；

（2）helpdesk 帮助桌面，浏览器模式；

（3）lookfor 返回包含指定关键词的项；

（4）demo 打开示例窗口。





MATLBA还提供了丰富的help命令，如表2-3所示，在命令窗口中输入相关命令就可以获得相关的帮助。

表2-3 MATLAB常用帮助命令

| 命令 | 内容 |
|---|---|
| help matfun | 矩阵函数—数值线性代数 |
| help general | 通用命令 |
| help graphics | 通用图形函数 |
| help elfun | 基本的数学函数 |
| help elmat | 基本矩阵和矩阵操作 |
| help control | 控制系统工具箱函数 |
| help datafun | 数据分析和傅里叶变换函数 |
| help ops | 操作符和特殊字符 |
| help polyfun | 多项式和内插函数 |
| help lang | 语言结构和调试 |
| help strfun | 字符串函数 |

## 2.5.1 直接使用help获得指令使用说明

可以直接使用help获得指令的使用说明，例如想准确知道所要求助的主题词，或指令名称，那么使用help是获得在线帮助的最简单有效的途径。

**例2-1** 使用正弦函数sin的在线求助。

**解：** 在命令窗口输入：

```
>> help sin
%输出为
 SIN      Sine.
    SIN(X) is the sine of the elements of X.
     See also asin, sind.
    Overloaded functions or methods (ones with the same name in other directories)
        help sym/sin.m
    Reference page in Help browser
        doc sin
```

## 2.5.2 直接使用help进行分类搜索

**例2-2** 使用help指令进行分类搜索，运行不带任何限定的help，得到分类名称明细表。

**解：** 在命令窗口输入：

```
>> help
%输出为
HELP topics

matlab\general          -   General purpose commands.
......              ......
slcontrol\slctrldemos -   (No table of contents file)
```

### 2.5.3 直接使用 help 获得具体子类指令说明

例 2-3 使用 help topic 指令形式获得具体子类的指令明细，如果用户想知道有关矩阵操作指令一览表，运行以下指令。

解：在命令窗口输入：

```
>> help polyfun
%输出为
Interpolation and polynomials.
  Data interpolation.
    pchip        - Piecewise cubic Hermite interpolating polynomial.
   ......          ......
    griddatan    - Data gridding and hyper-surface fitting (dimension >= 2).

  Spline interpolation.
    spline       - Cubic spline interpolation.
    ppval        - Evaluate piecewise polynomial.

  Geometric analysis.
    delaunay     - Delaunay triangulation.
     ......          ......
    polyarea     - Area of polygon.

  Polynomials.
    roots        - Find polynomial roots.
     ......          ......
    deconv       - Divide polynomials.
```




省略号由作者所加，表示出于节省篇幅的考虑被省略的内容。

### 2.5.4 使用 lookfor 指令

例 2-4 使用指令窗中的 lookfor 指令，查找包含积分这个关键词的所有指令。

解：在命令窗口输入：

```
>> lookfor integral
%输出为
ELLIPKE Complete elliptic integral.
EXPINT Exponential integral function.
   ......          ......
PEVINT Calculate integral of PEV
```




省略号由作者所加，表示出于节省篇幅的考虑被省略的内容。

## 2.6 小结

本章介绍 MATLAB 的启动、主要功能菜单、命令窗口、工作空间、文件管理和帮助管理等。通过阅读本章，读者对 MATLAB 的桌面操作环境能有一个比较全面的了解。



# 第3章 数组及其操作

在 MATLAB 内部任何数据类型，都是按照数组的形式进行存储和运算的。这里说的数组是广义的，它可以只有一个元素，也可以是一行或一列元素，还可能就是最普通的二维数组，抑或高维空间的多维数组；其元素也可以是任意数据类型，如数值型、逻辑型、字符串型，或元胞型等。

理解数组概念及其各种运算和操作，是学习 MATLAB 时的一个重要问题。本章以数值类型的一维和二维数组为例，重点讲解 MATLAB 中数组的概念和属性，以及创建、裁剪、寻址和运算等多种基本数组操作，这对于其他数据类型的数组，大部分都是通用的，因此读者要熟练掌握本章讲述的概念和操作函数。

## 3.1 MATLAB 中的数组

MATLAB 中数组可以说无处不在，任何变量在 MATLAB 中都是以数组形式存储和运算的。

按照数组元素个数和排列方式，MATLAB 中的数组可以分为：

（1）没有元素的空数组（empty array）；

（2）只有一个元素的标量（scalar），它实际上是一行一列的数组；

（3）只有一行或者一列元素的向量（vector），分别叫做行向量和列向量，也统称为一维数组；

（4）普通的具有多行多列元素的二维数组；

（5）超过二维的多维数组（具有行、列、页等多个维度，详见本书第 4 章）。

按照数组的存储方式，MATLAB 中的数组可以分为：普通数组和稀疏数组（常称为稀

疏矩阵）。稀疏矩阵适用于那些大部分元素为0，只有少部分非零元素的数组的存储，主要是为了提高数据存储和运算的效率。本书的第14章将介绍稀疏矩阵的部分内容。

## 3.2 数组的创建

MATLAB 中一般使用方括号（[]）、逗号（,）或空格，以及分号（;）来创建数组，方括号中给出数组的所有元素，同一行中的元素间用逗号或空格分隔，不同行之间用分号分隔。

### 3.2.1 创建空数组

空数组是 MATLAB 中特殊的数组，它不含有任何元素。空数组可以用于数组声明，数组清空，以及各种特殊的运算场合（如特殊的逻辑运算，见本书第8章）。

创建空数组很简单，只需要把变量赋值为空的方括号即可。

**例 3-1** 创建空数组 *A*。

**解：** 在命令窗口输入：

```
>> A=[]
%输出为
A =
     []
```

### 3.2.2 创建一维数组

一维数组包括行向量和列向量，是所有元素排列在一行或一列中的数组。实际上，一维数组可以看做二维数组在某一方向（行或列）尺寸退化为1的特殊形式。

创建一维行向量，只需要把所有用空格或逗号分隔的元素用方括号括起来即可；而创建一维列向量，则需要在方括号括起来的元素之间用分号分隔。不过，更常用的办法是用转置运算符（'），把行向量转置为列向量。

**例 3-2** 创建行向量和列向量。

**解：** 在命令窗口输入：

```
>> A=[1 2 3 4]
%输出为
A =
     1     2     3     4
>> B=[1;2;3;4]
%输出为
B =
     1
     2
     3
     4
```



很多时候要创建的一维数组实际上是个等差数列，这时候可以通过冒号（:）来创建。例如：

**Var**=start_val:step:stop_val

表示创建一个一维行向量 **Var**，它的第一个元素是 start_val，然后依次递增（step 为正）或递减（step 为负）step，直到向量中的最后一个元素与 stop_val 差的绝对值小于等于 step 的绝对值为止。当不指定 step 时，默认 step 等于 1。

和冒号功能类似的是 MATLAB 提供的 linspace 函数：

**Var**=linspace(start_val,stop_val,n)

表示创建一个一维行向量 **Var**，它的第一个元素是 start_val，最后一个元素是 stop_val，形成总共是 *n* 个元素的等差数列。不指定 *n* 时，默认 *n* 等于 100。要注意，这和冒号是不同的，冒号创建等差的一维数组时，stop_val 可能取不到。

一维列向量可以通过一维行向量的转置（'）得到。

**例 3-3** 创建一维等差数组。

**解**：在命令窗口输入：

```
>> A=1:4
A =
     1     2     3     4
>> B=1:2:4
B =
     1     3
>> D=linspace(1,4,5)
D =
    1.0000    1.7500    2.5000    3.2500    4.0000
```

类似于 linspace 函数，MATLAB 中还有创建等比一维数组的 logspace 函数：

**Var**=logspace(start_val,stop_val,n)

表示产生从 10^start_val 到 10^stop_val 包含 *n* 个元素的等比一维数组 **Var**。不指定 *n* 时，默认 *n* 等于 50。

**例 3-4** 创建一维等比数组。

**解**：在命令窗口输入：

```
>> A=logspace(0,log10(32),6)
A =
    1.0000    2.0000    4.0000    8.0000   16.0000   32.0000
```

创建一维数组可能用到：方括号、逗号或空格、分号、冒号、函数 linspace 和 logspace，以及转置符号（'）。

### 3.2.3 创建二维数组

常规创建二维数组的方法实际上和创建一维数组方法类似，就是综合运用方括号、逗号、空格，以及分号。方括号把所有元素括起来，不同行元素之间用分号间隔，同一行元素之间用逗号或者空格间隔，按照逐行排列的方式顺序书写每个元素。当然，在创建每一

行或列元素的时候，可以利用冒号和函数的方法，只是要特别注意创建二维数组时，要保证每一行（或每一列）具有相同数目的元素。

**例 3-5** 创建二维数组。

**解：** 在命令窗口输入：

```
>> A=[1 2 3;2 5 6;1 4 5]
A =
     1     2     3
     2     5     6
     1     4     5
>> B=[1:5;linspace(3,10,5);3 5 2 6 4]
B =
    1.0000    2.0000    3.0000    4.0000     5.0000
    3.0000    4.7500    6.5000    8.2500    10.0000
    3.0000    5.0000    2.0000    6.0000     4.0000
>> C=[[1:3]' [linspace(2,3,3)]' [3 5 6]']
C =
    1.0000    2.0000    3.0000
    2.0000    2.5000    5.0000
    3.0000    3.0000    6.0000
```

创建二维数组，也可以通过函数拼接一维数组，或者利用 MATLAB 内部函数直接创建特殊的二维数组，这些在本章后续内容中会逐步介绍。

## 3.3 数组属性

MATLAB 中提供了大量的函数，用于返回数组的各种属性，包括数组的排列结构、数组的尺寸大小、维度、数组数据类型，以及数组的内存占用情况等。

### 3.3.1 数组结构

数组的结构指的是数组中元素的排列方式。MATLAB 中的数组实际上就分为本章 3.1 节中介绍的几种。MATLAB 中提供了多种测试函数：

（1）isempty 检测某个数组是否是空数组；

（2）isscalar 检测某个数组是否是单元素的标量数组；

（3）isvector 检测某个数组是否是具有一行或一列元素的一维向量数组；

（4）issparse 检测某个数组是否是稀疏矩阵。

这些测试函数都是以 is 开头，然后紧跟检测的内容的关键字，它们的返回结果为逻辑类型，返回 1 表示测试符合条件，返回 0 表示测试不符合条件。关于稀疏矩阵的测试，本书第 14 章将进行讲解，这里只示例前几个数组结构的测试函数。

**例 3-6** 数组结构测试函数。

**解：** 在命令窗口输入：

```
>> A=32
```



```
A =
     32
>> isscalar(A)
ans =
     1
>> B=1:5
B =
     1     2     3     4     5
>> isvector(B)
ans =
     1
>> isempty(B)
ans =
     0
```

### 3.3.2　数组大小

数组大小是数组的最常用属性，它是指数组在每一个方向上具有的元素个数。例如，对于含有10个元素的一维行向量数组，则它在行的方向上（纵向）只有1个元素（1行），在列的方向上（横向）则有10个元素（10列）。

MATLAB中最常用的返回数组大小的是size函数。size函数有多种用法，对于一个$m$行$n$列的数组$A$，可以按以下两种方式使用size函数：

（1）$d$=size($A$)

将数组$A$的行列尺寸以一个行向量的形式返回给变量$d$，即$d=[m\ n]$；

（2）$[a,b]$=size($A$)

将数组$A$在行、列的方向的尺寸返回给$a$，$b$，即$a=m$，$b=n$。

length函数常用于返回一维数组的长度。

（1）当$A$是一维数组时，length($A$)返回此一维数组的元素个数；

（2）当$A$是普通二维数组时，length($A$)返回size($A$)得到的两个数中较大的那个。

在MATLAB中，空数组被默认为行的方向和列的方向尺寸都为0的数组，但如果自定义产生的多维空数组，则情况可能不同。

MATLAB中还有返回数组元素总个数的函数numel，对于$m$行$n$列的数组$A$，numel($A$)实际上返回$m*n$。

**例3-7**　数组大小。

**解**：在命令窗口输入：

```
>> A=[]
A =
     []
>> size(A)
ans =
     0     0
>> B=[1 2 3 4 5]
B =
     1     2     3     4     5
>> length(B)
```

```
ans =
     5
>> C=[1:5;2:6]
C =
     1     2     3     4     5
     2     3     4     5     6
>> size(C)
ans =
     2     5
>> length(C)
ans =
     5
>> numel(C)
ans =
    10
```

通过例 3-7 可以看出，MATLAB 通常把数组都按照普通的二维数组对待，即使是没有元素的空数组，也有行和列两个方向，只不过在这两个方向上它的尺寸都是零；而一维数组则是在行或者列中的一个方向的尺寸为 1；标量则在行和列两个方向上的尺寸都是 1。

### 3.3.3 数组维度

通俗一点讲数组维度，就是数组具有的方向。比如普通的二维数组，数组具有行的方向和列的方向，就是说数组具有两个方向，是一个二维数组。MATLAB 中还可以创建三维甚至更高维的数组。

对于空数组、标量和一维数组，MATLAB 还是当做普通二维数组对待的，因此它们都至少具有两个维度（至少具有行和列两个方向）。特别的，用空白方括号产生的空数组是当做二维数组对待的，但在高维数组中也有空数组的概念，这时候的空数组可以是只在任意一个维度上尺寸等于零的数组，相应地，此时的空数组就具有多个维度了。

MATLAB 中计算数组维度可以用函数 ndims。

ndims(*A*)返回结果实际上等于 length(size(*A*))。

例 3-8 数组维度。

**解**：在命令窗口输入：

```
>> B=2
B =
     2
>> ndims(B)
ans =
     2
>> C=1:5
C =
     1     2     3     4     5
>> ndims(C)
ans =
     2
```

通过例 3-8 可以看到，一般的非多维数组，在 MATLAB 中都是当做二维数组处理的。



### 3.3.4　数组数据类型

数组作为一种 MATLAB 的内部数据存储和运算结构，其元素可以是各种各样的数据类型（关于数据类型，可以参考本书第 5、6、7 章）。对应于不同的数据类型的元素，可以有数值数组（实数数组、浮点数值数组、整数数组等）、字符数组、元胞数组、字符串的元胞数组、结构体数组等，MATLAB 中提供了测试一个数组是否是这些类型的数组的测试函数，如表 3-1 所示。

表 3-1　数组数据类型测试函数

| 测试函数 | 说　明 |
| --- | --- |
| isnumeric | 测试一个数组是否是以数值型变量为元素的数组 |
| isreal | 测试一个数组是否是以实数数值型变量为元素的数组 |
| isfloat | 测试一个数组是否是以浮点数值型变量为元素的数组 |
| isinteger | 测试一个数组是否是以整数型变量为元素的数组 |
| islogical | 测试一个数组是否是以逻辑型变量为元素的数组 |
| ischar | 测试一个数组是否是以字符型变量为元素的数组 |
| isstruct | 测试一个数组是否是以结构体型变量为元素的数组 |
| iscell | 测试一个数组是否是以元胞型变量为元素的数组 |
| iscellstr | 测试一个数组是否是以结构体的元胞型变量为元素的数组 |

表 3-1 中，所有的测试函数同样都是以 is 开头，紧跟着一个测试内容关键字，它们的返回结果依然是逻辑类型，返回 0 表示不符合测试条件，返回 1 表示符合测试条件。

**例 3-9**　数组数据类型测试函数。

**解**：在命令窗口输入：

```
>> A=[1 2;3 5]
A =
     1     2
     3     5
>> isnumeric(A)
ans =
     1
>> isreal(A)
ans =
     1
>> isfloat(A)
ans =
     1
```

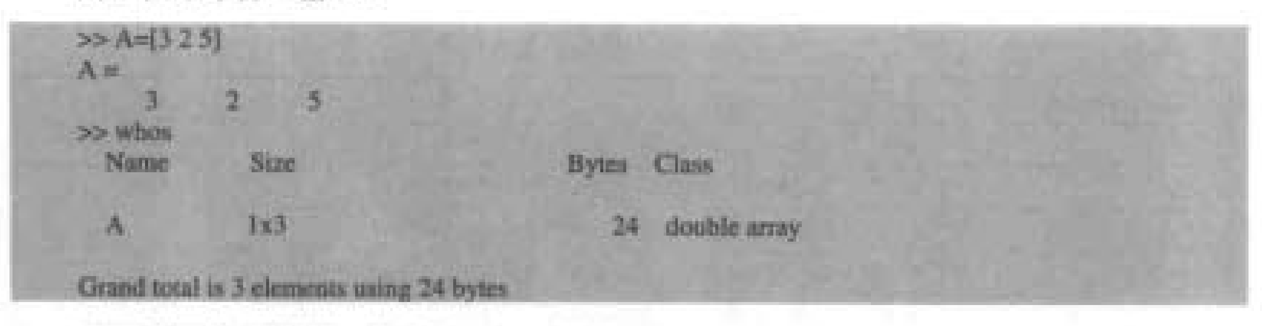
例 3-9 中用几个整数赋值的数组 $A$，实际上它的每一个元素都被当做双精度浮点数存储和运算（参考本书第 5 章）。因此，测试发现数组 $A$ 是一个实数数组、浮点数数组，而不是整数数组，更不是字符数组。这些测试函数在本书的后续章节还有所涉及。

### 3.3.5 数组的内存占用

了解数组的内存占用情况，对于优化 MATLAB 代码的性能是重要的。用户可以通过 whos 命令察看当前工作区中所有变量或指定变量的多种信息，包括变量名，数组大小，内存占用和数组元素的数据类型等。

例 3-10 数组的内存占用。

解：在命令窗口输入：



```
>> A=[3 2 5]
A =
     3     2     5
>> whos
  Name      Size                    Bytes  Class

  A         1x3                        24  double array

Grand total is 3 elements using 24 bytes
```

不同数据类型的数组的单个元素，内存占用是不一样的，用户可以通过 whos 命令计算各种数据类型的变量占用内存的情况，如例 3-10 中，1 行 3 列的双精度浮点型数组 ***A***，占用内存 24 字节，那么每一个双精度浮点型的元素就占用了 8 个字节的内存空间。通过简单的 whos 命令，用户就可以了解 MATLAB 中各种数据类型的内存占用情况，当然，读者也可以直接参考本书第 5 章学习不同数据类型变量的差别。

## 3.4 创建特殊数组

在矩阵代数领域，用户经常需要创建具有一定形式的特殊数组。MATLAB 提供了丰富的创建特殊数组的函数。

### 3.4.1 0−1 数组

顾名思义，0-1 数组就是所有元素不是 0 就是 1 的数组。在线性代数中，经常用到的 0-1 数组有：

（1）所有元素都为 0 的全 0 数组；

（2）所有元素都为 1 的全 1 数组；

（3）只有主对角线元素为 1，其他位置元素全部为 0 的单位数组；

（4）此外就是一般的 0-1 数组。

MATLAB 中，有专门的函数可以创建这些标准数组。

（1）zeros(*m*,*n*)

创建一个 *m* 行 *n* 列的全 0 数组，也可以用 zeros(size(***A***))创建一个和 ***A*** 具有相同大小的



全0数组。如果只指定一个数值，zeros(*m*)则创建一个 *m* 行 *m* 列的全0数组。

（2）ones(*m*,*n*)

ones(*m*,*n*)和 ones(size(***A***))则是创建 *m* 行 *n* 列，或者与 ***A*** 尺寸相同的全1数组。而ones(*m*)也是创建一个 *m* 行 *m* 列的全1数组。

（3）eye

用法和 zeros、ones 类似，不过创建的是指定大小的单位数组，即只有主对角线元素为1，其他元素全为0。

**例3-11** 创建0-1数组。

**解：**在命令窗口输入：

```
>> A=zeros(2)
A =
     0     0
     0     0
>> B=ones(2,4)
B =
     1     1     1     1
     1     1     1     1
>> C=eye(size(A))
C =
     1     0
     0     1
```

### 3.4.2 对角数组

在有些情况下，需要创建对角线元素为指定值，其他元素都为0的对角数组。这就要用到 diag 函数。

一般 diag 函数接收一个一维行向量数组为输入参数，将此向量的元素逐次排列在所指定的对角线上，其他位置则用0填充。

（1）diag(***v***)：创建一个对角数组，其主对角线元素依次对应于向量 ***v*** 的元素。

（2）diag(***v***,*k*)：创建一个对角数组，其第 *k* 条对角线元素对应于向量 ***v*** 的元素。当 *k* 大于0时，表示主对角线向右上角偏离 *k* 个元素位置的对角线；当 *k* 小于0时，表示主对角线向左下角偏离 *k* 个元素位置的对角线；当 *k* 等于0时，则和 diag(***v***)一样。

diag 函数也可以接受普通二维数组形式的输入参数，此时就不是创建对角数组了，而是从已知数组中提取对角元素组成一个一维数组。

（1）diag(***X***)提取二维数组 ***X*** 的主对角线元素组成一维数组。

（2）diag(***X***,*k*)提取二维数组 ***X*** 的第 *k* 条对角线元素组成的一维数组。

组合这两种用法，很容易产生已知数组 ***X*** 的指定对角线元素对应的对角数组，只需要通过组合命令 diag(diag(***X***,*m*),*n*)，就可以提取 ***X*** 的第 *m* 条对角线元素，产生与此对应的第 *n* 条对角线元素为提取的元素的对角数组。

**例 3-12** 创建对角数组。

**解：** 在命令窗口输入：

```
>> A=diag([1 2 3])
A =
     1     0     0
     0     2     0
     0     0     3
>> B=diag([1 2 3],2)
B =
     0     0     1     0     0
     0     0     0     2     0
     0     0     0     0     3
     0     0     0     0     0
     0     0     0     0     0
>> C=[3 5 1 2;2 4 5 6]
C =
     3     5     1     2
     2     4     5     6
>> diag(C)
ans =
     3
     4
>> D=diag(diag(C),-1)
D =
     0     0     0
     3     0     0
     0     4     0
```

这种组合使用两次 diag 函数产生对角数组的方法是常用的，读者需要加以掌握。

### 3.4.3 随机数组

在各种分析领域，随机数组都是很有用途的。MATLAB 中可以通过内部函数产生服从多种随机分布的随机数组，常用的有均匀分布和正态分布的随机数组。

（1）rand(*m*,*n*)可以产生 *m* 行 *n* 列的随机数组，其元素服从 0 到 1 的均匀分布；

（2）rand(size(*A*))产生和数组 ***A*** 具有相同大小的、元素服从 0 到 1 均匀分布的随机数组；

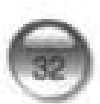
（3）rand(*m*)则产生 *m* 行 *m* 列的元素服从 0 到 1 均匀分布的随机数组。

randn 函数用于产生元素服从标准正态分布的随机数组，其用法和 rand 类似，此处不再赘述。

**例 3-13** 创建随机数组。

**解：** 在命令窗口输入：

```
>> A=rand(2)
A =
    0.9501    0.6068
    0.2311    0.4860
>> B=rand(2,4)
B =
    0.8913    0.4565    0.8214    0.6154
```





```
    0.7621    0.0185    0.4447    0.7919
>> C=randn(size(A))
C =
   -0.4326    0.1253
   -1.6656    0.2877
```

### 3.4.4 魔方数组

魔方数组也是一种比较常用的特殊数组，这种数组一定是正方形的（即行的方向的元素个数和列的方向的相等），而且每一行、每一列的元素之和都相等。

MATLAB 可以通过 magic(*n*)创建 *n* 行 *n* 列的魔方数组。

**例 3-14** 创建魔方数组。

**解**：在命令窗口输入：

```
>> magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2
```

读者可以自行验证，在例 3-14 中创建的魔方数组，其各行各列的算术和都是相等的。

利用 MATLAB 函数，除了可以创建这些常用的标准数组外，也可以创建许多专门应用领域常用的特殊数组，这将在本书第 14 章中作进一步介绍。

## 3.5 数组操作

前面讲解了 MATLAB 中数组的创建方法和基本属性，本节重点介绍在实际应用中最常用的一些数组操作方法。

### 3.5.1 数组的保存和装载

许多实际应用中的数组都是很庞大的，而且当操作步骤较多，不能在短期内完成，需要多次分时进行时，这些庞大的数组的保存和装载就是一个重要问题了，因为每次在进行操作前对数组进行声明和赋值，需要很庞大的输入工作量。一个好的解决方法是将数组保存在文件中，每次需要时进行装载。

MATLAB 中提供了内置的把变量保存在文件中的方法，最简单易用的是将数组变量保存为二进制的.mat 文件。用户可以通过 save 命令将工作区中指定的变量存储在.mat 文件中。

（1）save 命令的一般语法是：

save <filename> <var1> <var2>…<var*N*>

其作用是把 var1 var2…var*N* 指定的工作区变量存储在 filename 指定名称的.mat 文件中。通过 save 存储到.mat 文件中的数组变量，在使用前可以用 load 命令装载到工作区。

（2）load 命令的一般语法是：

load <filename> <var1> <var2>…<varN>

其作用是把当前目录下存储在 filename.mat 文件中的 var1 var2…varN 指定的变量装载到 MATLAB 工作区中。

关于 save 和 load 在数据保存和装载方面的更详细的内容，读者可以参考本书第 12 章。

### 3.5.2 数组索引和寻址

数组操作中最频繁遇到的就是对数组的某个具体位置上的元素进行访问和重新赋值，这涉及定位数组中元素位置，也就是数组索引和寻址的问题。

MATLAB 中数组元素的索引方式包括数字索引和逻辑索引两类。

**1. 数字索引方式**

MATLAB 中，普通二维数组元素的数字索引方式又可以分为两种：双下标（也叫全下标）索引和单下标索引。

双下标索引方式，顾名思义，就是用两个数字（自然数）来定位元素的位置。实际上就是用一个有序数对来表征元素位置，第一个数字指定元素所在的行位置，第二个数字指定元素所在的列。两个表示元素位置的索引数字之间用逗号分隔，并用圆括号括起来，紧跟在数组变量名后，就可以访问此数字索引指定的位置上的数组元素了。

例如，对于 3 行 2 列的数组 ***A***，***A***(3,1)表示数组 ***A*** 的第 3 行第 1 列的元素，***A***(1,2)表示数组 ***A*** 的第 1 行第 2 列的元素。

相应的，单下标索引方式就是用一个数字来定位数组元素。实际上，单下标索引和双下标索引是一一对应的，对一个已知尺寸的数组，任一个单下标索引数字都可以转换成确定的双下标索引。对于 $m$ 行 $n$ 列的数组 ***A***，***A***$(x,y)$实际上对应于 ***A***$((y-1)*m+x)$。

例如，对于 3 行 2 列的数组 ***A***，***A***(3,1)用单下标索引表示就是 ***A***(3)，***A***(1,2)用单下标索引表示就是 ***A***(4)。

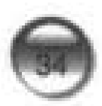
MATLAB 中单下标索引方式实际上采用了列元素优先的原则，即对于 $m$ 行 $n$ 列的数组 ***A***，第一列的元素的单下标索引依次为 ***A***(1)，***A***(2)，***A***(3)，…，***A***($m$)，第二列的元素的单下标索引依次为 ***A***($m$+1)，***A***($m$+2)，***A***($m$+3)，…，***A***(2$m$)，依此类推。

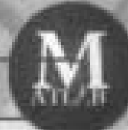
这两种数字索引方式中的数字索引也可以是一个数列，从而实现访问多个数组元素的目的，这通常可以通过应用冒号或一维数组来实现。

**例 3-15** 数组元素的索引与寻址。

**解**：在命令窗口输入：

```
>> A=[4 2 5 6;3 1 7 0;12 45 78 23]      %创建数组
A =
     4     2     5     6
     3     1     7     0
    12    45    78    23
>> A(2,3)          %双下标索引访问数组第 2 行第 3 列元素
```





```
ans =
     7
>> A(2)              %单下标索引访问数组第2个元素（即第2行第1列）
ans =
     3
>> A(7)              %单下标索引访问数组第7个元素（即第1行第3列）
ans =
     5
>> A(3:5)            %单下标索引访问数组第3到5位的元素（即第1列第3行和第2列第1、2行）
ans =
    12     2     1
>> A(2,1:4)          %双下标索引访问数组第2行，第1到4列的元素
ans =
     3     1     7     0
>> A([3,1],2)        %双下标索引访问数组第3、第1行，第2列的元素
ans =
    45
     2
>> A([3,1],[2,1])    %双下标索引访问数组第3、第1行，第2、第1列的元素
ans =
    45    12
     2     4
>> A(7)=100          %对数组第7个元素（即第1行第3列）重新赋值
A =
     4     2   100     6
     3     1     7     0
    12    45    78    23
```

通过例3-15可以看到，利用下标索引的方法，用户可以访问特定位置上的数组元素的值，或者对特定位置的数组元素重新赋值。

**2．单下标索引和双下标索引的转换**

单下标索引和双下标索引之间，可以通过MATLAB提供的函数进行转换。

把双下标索引转换为单下标索引，需要用sub2ind命令，其语法为：

IND = sub2ind(siz,*I*,*J*)

其中siz是一个1行2列的数组，指定转换数组的行列尺寸，一般可以用size(*A*)来表示；*I*和*J*分别是双下标索引中的两个数字；IND则为转换后的单下标数字。

把单下标索引转换为双下标索引，需要用ind2sub命令，其语法为：

[*I*,*J*] = ind2sub(siz,IND)

各变量意义同上。

**例3-16** 单-双下标转换。

**解**：在命令窗口输入：

```
>> A=rand(3,5)
A =
    0.4057    0.4103    0.3529    0.1389    0.6038
    0.9355    0.8936    0.8132    0.2028    0.2722
    0.9169    0.0579    0.0099    0.1987    0.1988
>> IND=sub2ind(size(A),2,4)
IND =
```

```
     11
>> A(IND)
ans =
     0.2028
>> [I,J]=ind2sub(size(A),13)
I =
     1
J =
     5
```

可以看到，sub2ind 函数和 ind2sub 函数实现了单-双下标的转换，需要注意的是，ind2sub 函数需要指定两个输出参数的接收变量，但由于 MATLAB 中小写字母 *i*，*j* 默认是用作虚数单位，因此最好是不用小写字母 *i*，*j* 来接收转换后的下标数字。

**3．逻辑索引方式**

除了这种双下标和单下标的数字索引外，MATLAB 中访问数组元素，还可以通过逻辑索引的方式。通常是通过比较关系运算产生一个满足比较关系的数组元素的索引数组（实际上是一个由 0，1 组成的逻辑数组），然后利用这个索引数组来访问原数组，并进行重新赋值等操作。

**例 3-17** 逻辑索引。

**解：** 在命令窗口输入：

```
>> A=rand(5)    %创建数组
A =
    0.0153    0.4186    0.8381    0.5028    0.1934
    0.7468    0.8462    0.0196    0.7095    0.6822
    0.4451    0.5252    0.6813    0.4289    0.3028
    0.9318    0.2026    0.3795    0.3046    0.5417
    0.4660    0.6721    0.8318    0.1897    0.1509
>> B=A>0.8     %通过比较关系运算产生逻辑索引（实际上是逻辑数组）
B =
     0     0     1     0     0
     0     1     0     0     0
     0     0     0     0     0
     1     0     0     0     0
     0     0     1     0     0
>> A(B)=0      %通过逻辑索引访问原数组元素，并重新赋值
A =
    0.0153    0.4186         0    0.5028    0.1934
    0.7468         0    0.0196    0.7095    0.6822
    0.4451    0.5252    0.6813    0.4289    0.3028
         0    0.2026    0.3795    0.3046    0.5417
    0.4660    0.6721         0    0.1897    0.1509
```

### 3.5.3 数组的扩展和裁剪

在许多操作过程中，需要对数组进行扩展或裁剪。数组扩展是指在超出数组现有尺寸的位置添加新元素；裁剪是指从现有数据中提取部分，产生一个新的小尺寸的数组。



1. 数组编辑器 Array Editor

数组编辑器是 MATLAB 提供的对数组进行编辑的交互式图形界面工具。双击 MATLAB 默认界面下工作区面板下的某一个变量，都能打开数组编辑器，从而进行数组元素的编辑。

数组编辑器界面类似于电子表格界面，每一个单元格就是一个数组元素。当单击超出数组当前尺寸的位置的单元格，并输入数据赋值时，实际上就是在该位置添加数组元素，即进行了数组的扩展操作，如图 3-1 所示。

图 3-1 数组编辑器中扩展数组

通过鼠标双击工作区面板下的 4 行 4 列的数组变量 *A*，打开了数组 *A* 的编辑器界面，然后在第 6 行第 6 列的位置单击单元格并输入数值，然后在其他位置单击鼠标或按下回车键，都可以使当前扩展操作即刻生效，如图 3-1 所示，数组 *A* 被扩展为 6 行 6 列的数组，原有元素不变，在第 6 行第 6 列的位置赋值为 3.12，其他扩展的位置上元素被默认赋值为 0。

通过数组编辑器也可以裁剪数组，这主要是对数组行、列的删除操作。需要通过鼠标右键菜单来实现。数组编辑器中单击某单元格后，单击鼠标右键，弹出如图 3-2 所示菜单。

图 3-2 数组编辑器右键菜单

在图 3-2 所示的菜单中，选择删除子菜单（Delete…），就可以指定删除当前数组中选

定位置元素所在的整行或者整列。删除子菜单的弹出窗口如图 3-3 所示，其中有多个选项，用户可以指定删除单元格数据、删除整行、删除整列。其中常用的是删除行列的操作，多次重复执行删除行、列操作，可以实现对数组的裁剪。



图 3-3　删除子菜单

图形用户界面的数组编辑器使用简单，但如果对数组的扩展或裁剪操作实际比较复杂时，通过数组编辑器实现是比较繁琐低效的。本节后面内容介绍通过 MATLAB 命令对数组的扩展和裁剪。

### 2. 数组扩展的 cat 函数

MATLAB 中可以通过 cat 系列函数将多个小尺寸数组按照指定的连接方式，组合成大尺寸的数组。这些函数包括：cat，horzcat 和 vertcat。

（1）cat 函数可以按照指定的方向将多个数组连接成大尺寸数组，其基本语法格式为：

C = cat(dim, ***A***1, ***A***2, ***A***3, ***A***4, …)，dim 用于指定连接方向。对于两个数组的连接，cat(1, ***A***, ***B***)实际上相当于[***A***; ***B***]，近似于把两个数组当做两个行元素连接；cat(2, ***A***, ***B***)实际上相当于[***A***, ***B***]，近似于把两个数组当做两个列元素连接。

（2）horzcat(***A***1, ***A***2, …)是水平方向连接数组，相当于 cat(2, ***A***1, ***A***2, …)；vertcat(***A***1, ***A***2, …)是垂直方向连接数组，相当于 cat(1, ***A***1, ***A***2, …)。

不管哪个连接函数，都必须保证被操作的数组可以被连接，即在某个方向上尺寸一致。如 horzcat 函数要求被连接的所有数组都具有相同的行数，而 vertcat 函数要求被连接的所有数组都具有相同的列数。

**例 3-18**　通过 cat 函数扩展数组。

**解：** 在命令窗口输入：

```
>> A=rand(3,5)
A =
    0.7468    0.4660    0.5252    0.8381    0.3795
    0.4451    0.4186    0.2026    0.0196    0.8318
    0.9318    0.8462    0.6721    0.6813    0.5028
>> B=eye(3)
B =
     1     0     0
     0     1     0
     0     0     1
>> C=magic(5)
C =
```



```
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>> cat(1,A,B)          %列数不同，不能垂直连接
??? Error using ==> cat
CAT arguments dimensions are not consistent.

>> cat(2,A,B)          %行数相同，可以水平连接
ans =
    0.7468    0.4660    0.5252    0.8381    0.3795    1.0000         0         0
    0.4451    0.4186    0.2026    0.0196    0.8318         0    1.0000         0
    0.9318    0.8462    0.6721    0.6813    0.5028         0         0    1.0000
>> cat(1,A,C)          %列数相同，可以垂直连接
ans =
    0.7468    0.4660    0.5252    0.8381    0.3795
    0.4451    0.4186    0.2026    0.0196    0.8318
    0.9318    0.8462    0.6721    0.6813    0.5028
   17.0000   24.0000    1.0000    8.0000   15.0000
   23.0000    5.0000    7.0000   14.0000   16.0000
    4.0000    6.0000   13.0000   20.0000   22.0000
   10.0000   12.0000   19.0000   21.0000    3.0000
   11.0000   18.0000   25.0000    2.0000    9.0000
>> cat(2,A,C)          %行数不同，不能水平连接
??? Error using ==> cat
CAT arguments dimensions are not consistent.

>> horzcat(A,B)
ans =
    0.7468    0.4660    0.5252    0.8381    0.3795    1.0000         0         0
    0.4451    0.4186    0.2026    0.0196    0.8318         0    1.0000         0
    0.9318    0.8462    0.6721    0.6813    0.5028         0         0    1.0000
>> horzcat(A,C)
??? Error using ==> horzcat
All matrices on a row in the bracketed expression must have the
 same number of rows.

>> vertzcat(A,C)
??? Undefined command/function 'vertzcat'.

>> vertcat(A,C)
ans =
    0.7468    0.4660    0.5252    0.8381    0.3795
    0.4451    0.4186    0.2026    0.0196    0.8318
    0.9318    0.8462    0.6721    0.6813    0.5028
   17.0000   24.0000    1.0000    8.0000   15.0000
   23.0000    5.0000    7.0000   14.0000   16.0000
    4.0000    6.0000   13.0000   20.0000   22.0000
   10.0000   12.0000   19.0000   21.0000    3.0000
   11.0000   18.0000   25.0000    2.0000    9.0000
```

### 3．块操作函数

MATLAB 中还有通过块操作实现数组扩展的函数。

（1）数组块状复制函数 repmat

repmat(*A*,*m*,*n*)可以将 *a* 行 *b* 列的元素 *A* 当做"单个元素"，扩展出 *m* 行 *n* 列个由此"单个元素"组成的扩展数组，实际上新产生的数组具有 *m***a* 行，*n***b* 列。

**例 3-19** 使用块状复制函数 repmat。

**解**：在命令窗口输入：

```
>> A=eye(2)
A =
     1     0
     0     1
>> repmat(A,2,2)
ans =
     1     0     1     0
     0     1     0     1
     1     0     1     0
     0     1     0     1
```

（2）对角块生成函数 blkdiag

blkdiag(*A*,*B*,…)将数组 *A*，*B* 等当做"单个元素"，安排在新数组的主对角线位置，其他位置用零数组块进行填充。

**例 3-20** 使用对角块生成函数 blkdiag。

**解**：在命令窗口输入：

```
>> A=eye(2)
A =
     1     0
     0     1
>> B=ones(2,3)
B =
     1     1     1
     1     1     1
>> blkdiag(A,B)
ans =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     1     1
     0     0     1     1     1
```

（3）块操作函数 kron

kron(*X*,*Y*)把数组 *Y* 当做一个"元素块"，先复制扩展出 size(*X*)规模的元素块，然后每一个块元素与 *X* 的相应位置的元素值相乘。

例如，对 2 行 3 列的数组 *X* 和任意数组 *Y*，kron(*X*,*Y*)返回的数组相当于[*X*(1,1)**Y* *X*(1,2)**Y* *X*(1,3)**Y*;X(2,1)**Y* *X*(2,2)**Y* *X*(2,3)*Y]。

**例 3-21** 使用块操作函数 kron。

**解**：在命令窗口输入：

```
>> A=[0 1 ;1  2]
B=magic(2)
C=kron(A,B)
A =
     0     1
```

```
     1     2
B =
     1     3
     4     2
C =
     0     0     1     3
     0     0     4     2
     1     3     2     6
     4     2     8     4
```

4．索引扩展

索引扩展是对数组进行扩展中最常用也最易用的方法。前面讲到索引寻址时，其中的数字索引有一定的范围限制，比如 $m$ 行 $n$ 列的数组 **A**，要索引寻址访问一个已有元素，通过单下标索引 **A**($a$)访问就要求 $a$<=$m$*$n$，因为 **A** 只有 $m$*$n$ 个元素，通过双下标索引 **A**($a$,$b$)访问就要求 $a$<=$m$，$b$<=$n$，因为 **A** 只有 $m$ 行 $n$ 列。

但索引扩展中使用的索引数字，就没有这些限制；相反，必然要用超出上述限制的索引数字，来指定当前数组尺寸外的一个位置，并对其进行赋值，以完成扩展操作。

通过索引扩展，一条语句只能增加一个元素，并同时在未指定的新添位置上默认赋值为 0。因此，要扩展多个元素就需要组合运用多条索引扩展语句，并且经常也要通过索引寻址修改特定位置上被默认赋值为 0 的元素。

**例 3-22** 索引扩展。

**解**：在命令窗口输入：

```
>> A=eye(3)
A =
     1     0     0
     0     1     0
     0     0     1
>> A(4,6)=25      %索引扩展
A =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     0     0    25
>> A(5,2)=3       %索引扩展
A =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     0     0    25
     0     3     0     0     0     0
>> A(5,5)=10      %索引寻址修改零元素
A =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     0     0    25
     0     3     0     0    10     0
```

通过例 3-22 可见，组合应用索引扩展和索引寻址重新赋值命令，在数组的索引扩展中

是经常会遇到的。

5．通过冒号操作符裁剪数组

相对于数组扩展这种放大操作，数组的裁剪就是产生新的子数组的缩小操作。从已知的大数据集中挑出一个子集合，作为新的操作对象，这在各种应用领域都是常见的。

MATLAB 中裁剪数组，最常用的就是冒号操作符，实际上，冒号操作符实现裁剪功能时，其意义和冒号用于创建一维数组的意义是一样的，都是实现一个递变效果。

例如从 100 行 100 列的数组 ***A*** 中挑选偶数行偶数列的元素，相对位置不变的组成 50 行 50 列的新数组 ***B***，只需要通过 ***B***=***A***(2:2:100,2:2:100)就可以实现。实际上这是通过数组数字索引实现了部分数据的访问。

更一般的裁剪语法是：

***B***=***A***([*a*1,*a*2,*a*3, …],[*b*1,*b*2,*b*3, …])

表示提取数组 ***A*** 的 *a*1，*a*2，*a*3，…等行，*b*1，*b*2，*b*3，…等列的元素组成子数组 ***B***。

此外，冒号还有一个特别的用法。当通过数字索引访问数组元素时，如果某一索引位置上不是用数字表示，而是用冒号代替，则表示这一索引位置可以取所有可以取到的值。例如对 5 行 3 列的数组 ***A***，***A***(3,:)表示取 ***A*** 的第三行所有元素（从第 1 列到第 3 列），***A***(:,2)表示取 ***A*** 的第 2 列的所有元素（从第 1 行到第 5 行）。

**例 3-23** 数组裁剪。

**解**：在命令窗口输入：

```
>> A=magic(8)
A =
    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1
>> A(1:2:5,3:7)  %提取数组 A 的第 1、3、5 行，3 到 7 列所有元素
ans =
     3    61    60     6     7
    46    20    21    43    42
    35    29    28    38    39
>> A(2,:)        %提取数组 A 的第 2 行所有元素
ans =
     9    55    54    12    13    51    50    16
>> A(:,3:2:7)    %提取数组 A 的第 3、5、7 列所有元素
ans =
     3    60     7
    54    13    50
    46    21    42
    27    36    31
    35    28    39
    22    45    18
    14    53    10
```



```
    59     4    63
>> A([3 2 1],[6 5 8])   %提取指定行列元素，注意新提取元素的顺序
ans =
    43    21    24
    51    13    16
     6    60    57
>> A(50:60)     %单下标索引裁减数组，提取第 50 到 60 号元素
ans =
    50    42    31    39    18    10    63    57    16    24    33
```

6．数组元素删除

通过部分的删除数组元素，也可以实现数组的裁剪。删除数组元素很简单，只需要对该位置元素赋值为空方括号（[]）即可，一般配合冒号，将数组的某些行、列元素删除。但要注意，进行删除时，索引结果必须是完整的行或完整的列，而不能是数组内部的块或单元格。

**例 3-24**　数组元素删除。

**解：** 在命令窗口输入：

```
>> A=magic(7)
A =
    30    39    48     1    10    19    28
    38    47     7     9    18    27    29
    46     6     8    17    26    35    37
     5    14    16    25    34    36    45
    13    15    24    33    42    44     4
    21    23    32    41    43     3    12
    22    31    40    49     2    11    20
>> A(1:3:8,:)=[]
A =
    38    47     7     9    18    27    29
    46     6     8    17    26    35    37
    13    15    24    33    42    44     4
    21    23    32    41    43     3    12
>> A(:,[3 5 6])=[]
A =
    38    47     9    29
    46     6    17    37
    13    15    33     4
    21    23    41    12
```

通过例 3-24 可见，数组元素的部分删除是直接在原始数组上进行的操作，在实际应用中，要考虑在数组元素删除前要不要先保存一个原始数组的拷贝，避免不小心造成对原始数据的破坏。另外，单独的一次删除操作只能删除某些行或某些列，因此一般需要通过两条语句才能实现行列两个方向的数组元素删除。

### 3.5.4　数组形状的改变

MATLAB 中有大量内部函数可以对数组进行改变形状的操作，包括数组转置，数组平移和旋转，以及数组尺寸的重新调整。

1．数组转置

MATLAB 中进行数组转置最简单的是通过转置操作符（'）。需要注意的是：

（1）对于有复数元素的数组，转置操作符（'）在变换数组形状的同时，也会将复数元素转化为其共轭复数。

（2）如果要对复数数组进行非共轭转置，可以通过点转置操作符（.'）实现。

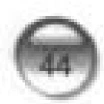
共轭和非共轭转置也可以通过 MATLAB 函数完成，transpose 实现非共轭转置，功能等同于点转置操作符（.'）；ctranspose 实现共轭转置，功能等同于转置操作符（'）。当然，这四种方法对于实数数组转置结果是一样的。

**例3-25** 数组转置。

**解**：在命令窗口输入：

```
>> A=rand(2,4)
A =
    0.9501    0.6068    0.8913    0.4565
    0.2311    0.4860    0.7621    0.0185
>> A'
ans =
    0.9501    0.2311
    0.6068    0.4860
    0.8913    0.7621
    0.4565    0.0185
>> B=[2-i,3+4i,2,5i;6+i,4-i,2i,7]
B =
  2.0000 - 1.0000i   3.0000 + 4.0000i   2.0000                  0 + 5.0000i
  6.0000 + 1.0000i   4.0000 - 1.0000i        0 + 2.0000i   7.0000
>> B'
ans =
  2.0000 + 1.0000i   6.0000 - 1.0000i
  3.0000 - 4.0000i   4.0000 + 1.0000i
  2.0000                  0 - 2.0000i
       0 - 5.0000i   7.0000
>> ctranspose(B)
ans =
  2.0000 + 1.0000i   6.0000 - 1.0000i
  3.0000 - 4.0000i   4.0000 + 1.0000i
  2.0000                  0 - 2.0000i
       0 - 5.0000i   7.0000
>> B.'
ans =
  2.0000 - 1.0000i   6.0000 + 1.0000i
  3.0000 + 4.0000i   4.0000 - 1.0000i
  2.0000                  0 + 2.0000i
       0 + 5.0000i   7.0000
>> transpose(B)
ans =
  2.0000 - 1.0000i   6.0000 + 1.0000i
  3.0000 + 4.0000i   4.0000 - 1.0000i
  2.0000                  0 + 2.0000i
       0 + 5.0000i   7.0000
```

实际使用中，由于操作符的简便性，经常会使用操作符而不是转置函数来实现转置。





但在复杂的嵌套运算中，转置函数可能是惟一的可用方法，所以，两类转置方式读者都要掌握。

### 2．数组翻转

MATLAB 中数组翻转的函数如表 3-2 所示。

表 3-2 数组翻转函数

| 函数及语法 | 说明 |
|---|---|
| fliplr(*A*) | 左右翻转数组 *A* |
| flipud(*A*) | 上下翻转数组 *A* |
| flipdim(*A*,*k*) | 按 *k* 指定的方向翻转数组<br>对于二维数组，*k*=1 相当于 flipud(*A*)；*k*=2 相当于 fliplr(*A*) |
| rot90(*A*,*k*) | 把 *A* 逆时针旋转 *k**90 度，*k* 不指定时默认取 1 |

**例 3-26** 数组翻转。

**解**：在命令窗口输入：

```
>> A=rand(4,6)
A =
    0.8214    0.9218    0.9355    0.0579    0.1389    0.2722
    0.4447    0.7382    0.9169    0.3529    0.2028    0.1988
    0.6154    0.1763    0.4103    0.8132    0.1987    0.0153
    0.7919    0.4057    0.8936    0.0099    0.6038    0.7468
>> flipud(A)
ans =
    0.7919    0.4057    0.8936    0.0099    0.6038    0.7468
    0.6154    0.1763    0.4103    0.8132    0.1987    0.0153
    0.4447    0.7382    0.9169    0.3529    0.2028    0.1988
    0.8214    0.9218    0.9355    0.0579    0.1389    0.2722
>> fliplr(A)
ans =
    0.2722    0.1389    0.0579    0.9355    0.9218    0.8214
    0.1988    0.2028    0.3529    0.9169    0.7382    0.4447
    0.0153    0.1987    0.8132    0.4103    0.1763    0.6154
    0.7468    0.6038    0.0099    0.8936    0.4057    0.7919
>> flipdim(A,2)
ans =
    0.2722    0.1389    0.0579    0.9355    0.9218    0.8214
    0.1988    0.2028    0.3529    0.9169    0.7382    0.4447
    0.0153    0.1987    0.8132    0.4103    0.1763    0.6154
    0.7468    0.6038    0.0099    0.8936    0.4057    0.7919
>> rot90(A)
ans =
    0.2722    0.1988    0.0153    0.7468
    0.1389    0.2028    0.1987    0.6038
    0.0579    0.3529    0.8132    0.0099
    0.9355    0.9169    0.4103    0.8936
    0.9218    0.7382    0.1763    0.4057
    0.8214    0.4447    0.6154    0.7919
>> rot90(A,2)
ans =
    0.7468    0.6038    0.0099    0.8936    0.4057    0.7919
```

```
    0.0153    0.1987    0.8132    0.4103    0.1763    0.6154
    0.1988    0.2028    0.3529    0.9169    0.7382    0.4447
    0.2722    0.1389    0.0579    0.9355    0.9218    0.8214
```

3．数组尺寸调整

改变数组形状，还有一个常用的函数是 reshape，它可以把已知数组改变成指定的行列尺寸。

对于 *m* 行 *n* 列的数组 ***A***，***B***=reshape(***A***,*a*,*b*)可以将其调整为 *a* 行 *b* 列的尺寸，并赋值给变量 ***B***。这里必须满足 *m***n*=*a***b*。在尺寸调整前后，两个数组的单下标索引不变，即 ***A***(*x*)必然等于 ***B***(*x*)，只要 *x* 是符合取值范围要求的单下标数字。也就是说，按照列优先原则把 ***A*** 和 ***B*** 的元素排成一列，那结果必然是一样的。

例 3-27 数组尺寸调整。

**解**：在命令窗口输入：

```
>> A=rand(3,4)
A =
    0.4451    0.4186    0.2026    0.0196
    0.9318    0.8462    0.6721    0.6813
    0.4660    0.5252    0.8381    0.3795
>> reshape(A,2,6)
ans =
    0.4451    0.4660    0.8462    0.2026    0.8381    0.6813
    0.9318    0.4186    0.5252    0.6721    0.0196    0.3795
>> reshape(A,6,2)
ans =
    0.4451    0.2026
    0.9318    0.6721
    0.4660    0.8381
    0.4186    0.0196
    0.8462    0.6813
    0.5252    0.3795
>> reshape(A,8,2)      %a*b 不等于 m*n 时会报错
??? Error using ==> reshape
To RESHAPE the number of elements must not change.
```

### 3.5.5 数组运算

本节介绍数组的各种数学运算。

1．数组-数组运算

最基本的就是数组和数组的加（+）、减（–）、乘（*）、乘方（^）等运算。要注意，数组的加、减，要求参与运算的两个数组具有相同的尺寸，而数组的乘法要求第一个数组的列数等于第二个数组的行数，乘方运算在指数 *n* 为自然数时相当于 *n* 次自乘，这要求数组具有相同的行数和列数。关于指数为其他情况的乘方，本节不作讨论，读者可以参考有关高等代数书籍。



**例 3-28**　使用数组-数组运算。

**解**：在命令窗口输入：

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> B=eye(4)
B =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> C=ones(4,2)
C =
     1     1
     1     1
     1     1
     1     1
>> A+B
ans =
    17     2     3    13
     5    12    10     8
     9     7     7    12
     4    14    15     2
>> B-A
ans =
   -15    -2    -3   -13
    -5   -10   -10    -8
    -9    -7    -5   -12
    -4   -14   -15     0
>> A*C
ans =
    34    34
    34    34
    34    34
    34    34
>> C*C'
ans =
     2     2     2     2
     2     2     2     2
     2     2     2     2
     2     2     2     2
>> (C*C')^3
ans =
   128   128   128   128
   128   128   128   128
   128   128   128   128
   128   128   128   128
```

数组除法实际上是乘法的逆运算，相当于参与运算的一个数组和另一个数组的逆（或伪逆）数组相乘。MATLAB 中数组除法有左除（/）和右除（\）两种：

（1）***A*/*B*** 相当于 ***A****inv(***B***)或 ***A****pinv(***B***)。

（2）**A\B** 相当于 inv(**A**)***B** 或 pinv(**A**)***B**。

其中 inv 是数组求逆函数，仅适用于行列数相同的方形数组（线性代数中，称为方阵）；pinv 是求数组广义逆的函数。关于逆矩阵和广义逆矩阵的知识，请读者参考有关的高等代数的书籍。

**例 3-29** 使用数组除法。

**解**：在命令窗口输入：

```
>> A=[3 5 6;2 1 4;2 5 6]
A =
     3     5     6
     2     1     4
     2     5     6
>> B=randn(3)
B =
   -0.4326    0.2877    1.1892
   -1.6656   -1.1465   -0.0376
    0.1253    1.1909    0.3273
>> A/B
ans =
    4.9369   -3.0803    0.0406
    3.9586   -2.4124   -2.4389
    4.7620   -2.3806    0.7564
>> A*inv(B)
ans =
    4.9369   -3.0803    0.0406
    3.9586   -2.4124   -2.4389
    4.7620   -2.3806    0.7564
>> A\B
ans =
   -0.5579   -0.9032    0.8619
    0.5902    0.5735    0.3559
   -0.2850    0.0216   -0.5293
>> pinv(A)*B
ans =
   -0.5579   -0.9032    0.8619
    0.5902    0.5735    0.3559
   -0.2850    0.0216   -0.5293
```

## 2. 点运算

前面讲到的数组乘、除、乘方运算，都是专门针对数组定义的运算。有些情况下，用户可能希望对两个尺寸相同的数组进行元素对元素的乘、除，或者对数组的逐个元素进行乘方，这可以通过点运算实现。

**A.*B**，就可以实现两个同样尺寸的数组 **A** 和数组 **B** 对应元素的乘法，同样的，**A./B** 或 **A.\B** 实现元素对元素的除法，**A.^n** 实现对逐个元素的乘方。

**例 3-30** 使用点运算。

**解**：在命令窗口输入：

```
>> A=magic(4)
A =
    16     2     3    13
```



```
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> B=ones(4)+4*eye(4)
B =
     5     1     1     1
     1     5     1     1
     1     1     5     1
     1     1     1     5
>> A.*B
ans =
    80     2     3    13
     5    55    10     8
     9     7    30    12
     4    14    15     5
>> B.*A    %对应元素的乘法，因此和 A.*B 结果一样
ans =
    80     2     3    13
     5    55    10     8
     9     7    30    12
     4    14    15     5
>> B.^3
ans =
   125     1     1     1
     1   125     1     1
     1     1   125     1
     1     1     1   125
>> A.\B    %以 A 的各个元素为分母，B 的各个元素为分子，逐个元素作除法
ans =
    0.3125    0.5000    0.3333    0.0769
    0.2000    0.4545    0.1000    0.1250
    0.1111    0.1429    0.8333    0.0833
    0.2500    0.0714    0.0667    5.0000
>> A./B    %以 A 的各个元素为分子，B 的各个元素为分母，逐个元素作除法
ans =
    3.2000    2.0000    3.0000   13.0000
    5.0000    2.2000   10.0000    8.0000
    9.0000    7.0000    1.2000   12.0000
    4.0000   14.0000   15.0000    0.2000
```

特别要强调的是，许多 MATLAB 内置的运算函数，如 sqrt，exp，log，sin，cos 等，都只能对数组进行逐个元素的相应运算。至于专门的数组的开方、指数等运算，都有专门的数组运算函数。

### 3. 专门针对数组的运算函数

MATLAB 中，专门针对数组的运算函数一般末尾都以 *m* 结尾（*m* 代表 matirx），如 sqrtm、expm、logm 等，这些运算都是特别定义的数组运算，不同于针对单个数值的常规数学运算。这几个函数都要求参与运算的数组是行数和列数相等的方形数组。具体的运算方法请参考高等代数方面的书籍。

**例 3-31** 使用数组运算函数。

**解**：在命令窗口输入：

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> sqrt(A)
ans =
    4.0000    1.4142    1.7321    3.6056
    2.2361    3.3166    3.1623    2.8284
    3.0000    2.6458    2.4495    3.4641
    2.0000    3.7417    3.8730    1.0000
>> sqrtm(A)
ans =
   3.7584 - 0.2071i  -0.2271 + 0.4886i   0.3887 + 0.7700i   1.9110 - 1.0514i
   0.2745 - 0.0130i   2.3243 + 0.0306i   2.0076 + 0.0483i   1.2246 - 0.0659i
   1.3918 - 0.2331i   1.5060 + 0.5498i   1.4884 + 0.8666i   1.4447 - 1.1833i
   0.4063 + 0.4533i   2.2277 - 1.0691i   1.9463 - 1.6848i   1.2506 + 2.3006i
>> exp(A)
ans =
  1.0e+006 *
    8.8861    0.0000    0.0000    0.4424
    0.0001    0.0599    0.0220    0.0030
    0.0081    0.0011    0.0004    0.1628
    0.0001    1.2026    3.2690    0.0000
>> expm(A)
ans =
  1.0e+014 *
    1.4587    1.4587    1.4587    1.4587
    1.4587    1.4587    1.4587    1.4587
    1.4587    1.4587    1.4587    1.4587
    1.4587    1.4587    1.4587    1.4587
```

### 3.5.6 数组查找

MATLAB 中数组查找只有一个函数 find，它能够查找数组中的非零元素并返回其下标索引。find 配合各种关系运算和逻辑运算，能够实现很多查找功能。

find 函数有两种语法形式：

（1）*a*=find(***A***)返回数组 ***A*** 中非零元素的单下标索引；

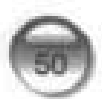
（2）[*a*,*b*]=find(***A***)返回数组 ***A*** 中非零元素的双下标索引方式。

实际应用中，经常是通过多重逻辑嵌套产生逻辑数组，判断数组元素是否符合某种比较关系，然后用 find 函数查找这个逻辑数组中的非零元素，返回符合比较关系的元素的索引，从而实现元素访问。find 用于产生索引数组，过渡实现最终的索引访问，因此经常不需要直接指定 find 函数的返回值。

例 3-32　使用数组查找函数 find。

**解：** 在命令窗口输入：

```
>> A=rand(3,5)
A =
    0.6822    0.1509    0.8600    0.4966    0.6449
    0.3028    0.6979    0.8537    0.8998    0.8180
```





```
    0.5417    0.3784    0.5936    0.8216    0.6602
>> A>0.3
ans =
     1     0     1     1     1
     1     1     1     1     1
     1     1     1     1     1
>> A<0.5
ans =
     0     1     0     1     0
     1     0     0     0     0
     0     1     0     0     0
>> (A>0.3)&(A<0.5)        逻辑嵌套产生符合多个比较关系的逻辑数组
ans =
     0     0     0     1     0
     1     0     0     0     0
     0     1     0     0     0
>> find((A>0.3)&(A<0.5))  %find 逻辑数组中的非零元素，返回符合关系的元素索引
ans =
     2
     6
    10
>> A(find((A>0.3)&(A<0.5)))    %实现元素访问
ans =
    0.3028
    0.3784
    0.4966
```

例 3-32 一步一步地示例了 find 的最常见用法的具体使用过程。首先通过 rand 函数创建了待操作的随机数组 ***A***，然后通过比较运算 ***A***>0.3 和 ***A***<0.5 返回分别满足某一比较关系的逻辑数组，在这些逻辑数组中，1 代表该位置元素符合比较关系，0 则代表不符合比较关系，然后通过逻辑运算（&）可以产生同时满足两个比较关系的逻辑数组，find 操作这个逻辑数组，返回数组中非零元素的下标索引（本例中返回单下标索引），实际上就是返回原数组中符合两个比较关系的元素的位置索引，然后利用 find 返回的下标索引就可以寻址访问原来数组中符合比较关系的目标元素了。

## 3.5.7 数组排序

数组排序也是常用的数组操作，经常用在各种数据分析和处理中，MATLAB 中的排序函数是 sort。

sort 函数可以对数组按照升序或降序进行排列，并返回排序后的元素在原数组中的索引位置，sort 函数有多种应用语法格式，都有重要的应用，见表 3-3。

表 3-3 sort 函数的各种语法格式

| 函数语法 | 说 明 |
|---|---|
| *B*=sort(*A*) | 对一维或二维数组进行升序排序，并返回排序后的数组<br>当 *A* 为二维数组时，则是对数组的每一列进行排序 |
| *B*=sort(*A*,dim) | 对数组按指定的方向进行升序排序<br>dim=1 表示对每一列排序，dim=2 表示对每一行排序 |

续表

| 函数语法 | 说　明 |
| --- | --- |
| B=sort(A,dim,mode) | mode 指定排序模式<br>mode 为"ascend"时，进行升序排序；mode 为"descend"时，进行降序排序 |
| [B,IX]=sort(A,…) | IX 为排序后各元素在原数组中的行位置或列位置的索引 |

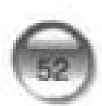
可以看到，sort 都是对单独的一行或一列元素进行排序，即使对于二维数组，也是单独对每一行每一列进行排序，因此返回的索引只是单下标形式，表征排序后元素在原来行（或列）中的位置。

**例 3-33** 数组排序。

**解**：在命令窗口输入：

```
>> A=rand(1,8)
A =
    0.0648    0.9883    0.5828    0.4235    0.5155    0.3340    0.4329    0.2259
>> sort(A)  %按默认的升序方式排序
ans =
    0.0648    0.2259    0.3340    0.4235    0.4329    0.5155    0.5828    0.9883
>> [B,I]=sort(A,'descend')  %降序排序并返回索引
B =
    0.9883    0.5828    0.5155    0.4329    0.4235    0.3340    0.2259    0.0648
I =
     2     3     5     7     4     6     8     1
>> A(I)  %通过索引也可以产生降序排序的数组
ans =
    0.9883    0.5828    0.5155    0.4329    0.4235    0.3340    0.2259    0.0648
>> C=rand(3,6)
C =
    0.5798    0.6405    0.7833    0.5678    0.6029    0.3050
    0.7604    0.2091    0.6808    0.7942    0.0503    0.8744
    0.5298    0.3798    0.4611    0.0592    0.4154    0.0150
>> sort(C)  %对每一列进行升序排序
ans =
    0.5298    0.2091    0.4611    0.0592    0.0503    0.0150
    0.5798    0.3798    0.6808    0.5678    0.4154    0.3050
    0.7604    0.6405    0.7833    0.7942    0.6029    0.8744
>> [D,I]=sort(C,2)          %指定对每一行进行升序排序，并返回排序后各元素的列号
D =
    0.3050    0.5678    0.5798    0.6029    0.6405    0.7833
    0.0503    0.2091    0.6808    0.7604    0.7942    0.8744
    0.0150    0.0592    0.3798    0.4154    0.4611    0.5298
I =
     6     4     1     5     2     3
     5     2     3     1     4     6
     6     4     2     5     3     1
>> C(I)  %对二维数组不能通过 sort 返回的单下标索引产生排序的数组
ans =
    0.3798    0.6405    0.5798    0.2091    0.7604    0.5298
    0.2091    0.7604    0.5298    0.5798    0.6405    0.3798
    0.3798    0.6405    0.7604    0.2091    0.5298    0.5798
```

通过例 3-33 可见，数组排序函数 sort 返回的索引，是表示了在排序方向上排序后元素



在原数组中的位置，对于一维数组，这就是其单下标索引，但对二维数组，这只是双下标索引中的一个分量，因此不能简单地通过这个返回的索引值寻址产生排序的二维数组。

当然，利用这个索引结果，通过复杂一点的方法也可以得到排序数组，如例 3-33 中，就可以通过 ***D***=[***C***(1,***I***(1,:)); ***C***(2,***I***(2,:)); ***C***(2,***I***(2,:))]来产生排序数组，这种索引访问，一般只用在对部分数据的处理上。

## 3.6 小结

数组是 MATLAB 中各种变量存储和运算的通用数据结构。本章从对 MATLAB 中的数组进行分类概述入手，重点讲述了数组的创建、数组的属性和多种数组操作方法，这些内容是学习 MATLAB 必须熟练掌握的。对这些基本函数的深入的理解和熟练的组合应用，会大大提高使用 MATLAB 的效率，因此，读者对本章中的所有函数都要仔细体会，熟练掌握。



# 第 4 章

# 多维数组及其操作

MATLAB 中把超过两维的数组称为多维数组，多维数组实际上是一般的二维数组的扩展。本章讲述 MATLAB 中多维数组的创建和操作。

## 4.1 多维数组

对于二维数组，习惯把第一维称为行，第二维称为列，这样，二维数组就是一个行-列确定的一个“数组面”。多维数组实际上是在多维空间中对这个数组面的扩展，例如三维数组，就是在行、列这两个维度之外增加了第三个方向的维度，一般把这第三个维度称为页，三维数组实际上可以看做行-列-页确定的一个“长方体”。同样地，更高维的数组就是用后面添加的维度来确定这个页而已。本节主要介绍 MATLAB 中创建多维数组的方法和获取多维数组属性的函数。

### 4.1.1 多维数组的创建

MATLAB 中创建多维数组有 3 种方法：

（1）通过指定索引把二维数组扩展成多维数组；

（2）采用 MATLAB 的内联函数创建；

（3）采用 cat 函数进行连接创建。

下面分别对这 3 种方法进行介绍。

#### 1．通过指定索引把二维数组扩展成多维数组

通过二维数组扩展创建三维数组，就是先创建三维数组中每一页上的二维数组元素，



然后将它们赋值到三维数组第三维的指定位置上。类似的方法，可以通过对二维数组的扩展创建更高维的数组。需要注意的是，对于没有指定赋值的维度上的元素，MATLAB 默认赋值为 0。

**例 4-1** 通过二维数组扩展创建多维数组。

二维数组 *A* 是一个 3×3 的数组，通过增加一个 3×3 的数组，把它扩展成一个 3×3×2 的多维数组。

**解**：在命令窗口输入：

```
>> A = [5 7 8; 0 1 9; 4 3 6];
>> A(:,:,2) = [1 0 4; 3 5 6; 9 8 7]
A(:,:,1) =
     5     7     8
     0     1     9
     4     3     6
A(:,:,2) =
     1     0     4
     3     5     6
     9     8     7
```

### 2．采用 MATLAB 的内联函数创建

第 3 章中介绍了 MATLAB 中许多创建二维数组的内联函数，如 ones，zeros，rand，randn 等（见表 4-1），通过改变这些函数的输入参数也可以创建多维数组。需要注意的是，当多维数组的任何一个维度被指定为 0，都代表创建了一个多维空数组。

表 4-1　MATLAB 中用来创建多维数组的内联函数及其语法

| 函数语法 | 功　能 |
|---|---|
| ones(*d1*,*d2*,*d3*,…) | 生成 *d1***d2***d3**…的多维全 1 数组 |
| ones(size(*A*)) | 生成与数组 *A* 同样尺寸的全 1 数组 |
| zeros(*d1*,*d2*,*d3*,…) | 生成 *d1***d2***d3**…的多维全 0 数组 |
| zeros(size(*A*)) | 生成与数组 *A* 同样尺寸的全 0 数组 |
| rand(*d1*,*d2*,*d3*,…) | 生成 *d1***d2***d3**…的多维数组，数组元素服从[0,1]均匀分布 |
| rand(size(*A*)) | 生成与数组 *A* 同样尺寸数组，数组元素服从[0,1]均匀分布 |
| randn(*d1*,*d2*,*d3*,…) | 生成 *d1***d2***d3**…的多维数组，数组元素服从 N(0,1)标准正态分布 |
| randn(size(*A*)) | 生成与数组 *A* 同样尺寸的数组，数组元素服从 N(0,1)标准正态分布 |
| repmat(*m*,[*d1*,*d2*,*d3*…]) | 生成 *d1***d2***d3*…的多维数组，数组元素为 *m*<br>注：*m* 也可以是一维向量或者二维数组，这样就是对数组的块操作复制 |

**例 4-2** 用 MATLAB 的内联函数创建多维数组。

*A* 是一个随机生成的 3×3×2 的多维数组，*B* 是一个 3×4×2 的多维数组，它的元素全部为 5。

**解**：在命令窗口输入：

```
>> A = randn(4,3,2)
A(:,:,1) =
   -0.4326   -1.1465    0.3273
   -1.6656    1.1909    0.1746
```

```
    0.1253     1.1892    -0.1867
    0.2877    -0.0376     0.7258
A(:,:,2) =
   -0.5883     1.0668     0.2944
    2.1832     0.0593    -1.3362
   -0.1364    -0.0956     0.7143
    0.1139    -0.8323     1.6236
>> B = repmat(5, [3 4 2])
B(:,:,1) =
     5     5     5     5
     5     5     5     5
     5     5     5     5
B(:,:,2) =
     5     5     5     5
     5     5     5     5
     5     5     5     5
```

3. 采用 cat 函数进行连接创建

除此之外，MATLAB 中还专门提供了 cat 函数用于创建多维数组。cat 函数可以把几个预先赋值好的数组或者新建立的数组按照某一维度连接起来，创建一个多维数组。当某些中间维度没有被指定时，cat 函数默认把它们指定为 1。

例 4-3 用 cat 函数创建多维数组。

**解**：在命令窗口输入：

```
>> A = cat(3, [9 2; 6 5], [7 1; 8 4]); %在页方向连接上两个新建立的二维数组创建三维数组
>> B = cat(3, [3 5; 0 1], [5 6; 2 1]); %同上，用 cat 在第三维方向上连接，创建三维数组
>> C = cat(5, A, B, cat(3, [1 2; 3 4], [4 3; 2 1]))    %在第五维上连接三个三维数组
C(:,:,1,1,1) =
     9     2
     6     5
C(:,:,2,1,1) =
     7     1
     8     4
C(:,:,1,1,2) =
     3     5
     0     1
C(:,:,2,1,2) =
     5     6
     2     1
C(:,:,1,1,3) =
     1     2
     3     4
C(:,:,2,1,3) =
     4     3
     2     1
```



### 4.1.2　多维数组的属性

MATLAB 中提供了多个函数，可以获得多维数组的尺寸、维度、占用内存和数据类型等多种属性。

表 4-2　MATLAB 中获取多维数组属性的函数

| 数组属性 | 函数用法 | 函数功能 |
| --- | --- | --- |
| 尺寸 | size(*C*) | 按照行-列-页-…的顺序，返回数组 *C* 的每一维上的大小 |
| 维度 | ndims(*C*) | 返回数组 *C* 具有的维度值 |
| 内存占用/数据类型等 | whos | 返回当前工作区中的各个变量的详细信息 |

**例 4-4**　通过 MATLAB 函数获取多维数组的属性。

**解**：在命令窗口输入：

```
>> A = cat(4, [9 2; 6 5], [7 1; 8 4]);
>> size(A)    %获取数组 A 的尺寸属性
ans =
     2     2     1     2
>> ndims(A)      %获取数组 A 的维度属性
ans =
     4
>> whos    %显示当前工作区中各个变量的名称、尺寸、内存占用等信息
  Name         Size                    Bytes  Class

  A            4-D                        64  double array
  ans          1x1                         8  double array

Grand total is 9 elements using 72 bytes
```

## 4.2　多维数组的操作

和二维数组类似，MATLAB 中也有大量对多维数组进行索引、重排和计算的函数。

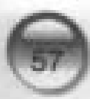
### 4.2.1　多维数组的索引

MATLAB 中索引多维数组的方法包括多下标索引和单下标索引。

对于 $n$ 维数组可以用 $n$ 个下标索引访问到一个特定位置的元素，而使用数组或者冒号来代表其中某一维，则可以访问指定位置的多个元素。单下标索引方法则是只通过一个下标来定位多维数组中某个元素的位置，只要注意到 MATLAB 中是按照行-列-页-…优先级逐渐降低的顺序把多维数组的所有元素线性存储起来，就可以知道一个特定的单下标对应的多维下标位置了。

**例 4-5**　多维数组的索引访问，其中 *A* 是一个随机生成的 4×5×3 的多维数组。

**解**：在命令窗口输入：



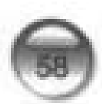
```
>> A=randn(4,5,3)
A(:,:,1) =
    0.1286   -0.2624    0.0112   -0.9898    1.1380
    0.6565   -1.2132   -0.6451    1.3396   -0.6841
   -1.1678   -1.3194    0.8057    0.2895   -1.2919
   -0.4606    0.9312    0.2316    1.4789   -0.0729
A(:,:,2) =
   -0.3306   -0.5465   -0.8542    0.4853   -0.0793
   -0.8436   -0.8468   -1.2013   -0.5955    1.5352
    0.4978   -0.2463   -0.1199   -0.1497   -0.6065
    1.4885    0.6630   -0.0653   -0.4348   -1.3474
A(:,:,3) =
    0.4694    0.5354    0.1326   -0.0787    0.2888
   -0.9036    0.5529    1.5929   -0.6817   -0.4293
    0.0359   -0.2037    1.0184   -1.0246    0.0558
   -0.6275   -2.0543   -1.5804   -1.2344   -0.3679
>> A(3,2,2)   %访问 A 的第 3 行第 2 列第 2 页的元素
ans =
   -0.2463
>> A(3,2,:)   %访问 A 的第 3 行第 2 列的所有页的元素
ans(:,:,1) =
   -1.3194
ans(:,:,2) =
   -0.2463
ans(:,:,3) =
   -0.2037
>> A(:,:,[1 2]) %访问 A 的第 1 页和第 2 页的元素
ans(:,:,1) =
    0.1286   -0.2624    0.0112   -0.9898    1.1380
    0.6565   -1.2132   -0.6451    1.3396   -0.6841
   -1.1678   -1.3194    0.8057    0.2895   -1.2919
   -0.4606    0.9312    0.2316    1.4789   -0.0729
ans(:,:,2) =
   -0.3306   -0.5465   -0.8542    0.4853   -0.0793
   -0.8436   -0.8468   -1.2013   -0.5955    1.5352
    0.4978   -0.2463   -0.1199   -0.1497   -0.6065
    1.4885    0.6630   -0.0653   -0.4348   -1.3474
>> A(27)
ans =
   -0.2463
```

例 4-5 中，***A***(27)是通过单下标索引来访问 4×5×3 的多维数组 ***A*** 的元素，因为多维数组 ***A*** 有 3 页，每一页有 4×5=20 个元素，所以第 27 个元素在第二页上，而每一页上行方向上有 4 个元素，根据行-列-页的优先原则，第 27 个元素代表的就是第二页上第二列的第三行，即 ***A***(27)相当于 ***A***(3,2,2)。

### 4.2.2 多维数组的维度操作

多维数组的维度操作包括对多维数组形状的重排和维度的重新排序。

reshape 函数可以改变多维数组的形状，但操作前后 MATLAB 中按照行-列-页-…优先级对多维数组进行线性存储的方式不变。许多多维数组在某一维度上只有一个元素，可以





利用函数 squeeze 来清除这种单值维度。

**例 4-6**　用 reshape 函数改变多维数组的形状。

**解**：在命令窗口输入：

```
>> A=randn(3,4,1,2)    %创建数组多维 A,它的维数是 3*4*1*2
A(:,:,1,1) =
    1.9574   -0.3398    1.1902   -0.6014
    0.5045   -1.1398   -1.1162    0.5512
    1.8645   -0.2111    0.6353   -1.0998
A(:,:,1,2) =
    0.0860    0.4620   -0.6313    1.0556
   -2.0046   -0.3210   -2.3252   -0.1132
   -0.4931    1.2366   -1.2316    0.3792
>> reshape(A,[2 4 1 3])
ans(:,:,1,1) =
    1.9574    1.8645   -1.1398    1.1902
    0.5045   -0.3398   -0.2111   -1.1162
ans(:,:,1,2) =
    0.6353    0.5512    0.0860   -0.4931
   -0.6014   -1.0998   -2.0046    0.4620
ans(:,:,1,3) =
   -0.3210   -0.6313   -1.2316   -0.1132
    1.2366   -2.3252    1.0556    0.3792
>> size(A)
ans =
     3     4     1     2
>> B=squeeze(A);
>> size(B)
ans =
     3     4     2
```

permute 函数可以按照指定的顺序重新定义多维数组的维度顺序，需要注意的是，permute 重新定义后的多维数组是把原来在某一维度上的所有元素移动到新的维度上，这会改变多维数组线性存储的位置，和 reshape 是不同的。ipermute 可以被看做是 permute 的逆函数，当 ***B***=permute(***A***,dims)时，ipermute(***B***,dims)刚好返回多维数组 ***A***。

**例 4-7**　对多维数组维度的重新排序。

**解**：在命令窗口输入：

```
>> A=randn(3,3,2)
A(:,:,1) =
    0.9442   -0.7043    1.5210
   -2.1204   -1.0181   -0.0384
   -0.6447   -0.1821    1.2274
A(:,:,2) =
   -0.6962    0.5869    0.6682
    0.0075   -0.2512   -0.0783
   -0.7829    0.4801    0.8892
>> B=permute(A,[3 1 2])
B(:,:,1) =
    0.9442   -2.1204   -0.6447
   -0.6962    0.0075   -0.7829
B(:,:,2) =
```

```
   -0.7043   -1.0181   -0.1821
    0.5869   -0.2512    0.4801
B(:,:,3) =
    1.5210   -0.0384    1.2274
    0.6682   -0.0783    0.8892
>> ipermute(B,[3 1 2])
ans(:,:,1) =
    0.9442   -0.7043    1.5210
   -2.1204   -1.0181   -0.0384
   -0.6447   -0.1821    1.2274
ans(:,:,2) =
   -0.6962    0.5869    0.6682
    0.0075   -0.2512   -0.0783
   -0.7829    0.4801    0.8892
```

### 4.2.3 多维数组参与数学计算

多维数组参与数学计算，可以针对某一维度的向量，也可以针对单个元素，或者针对某一特定页面上的二维数组。

（1）sum，mean 等函数可以对多维数组中第 1 个不为 1 的维度上的向量进行计算；

（2）sin，cos 等函数则对多维数组中的每一个单独元素进行计算；

（3）eig 等针对二维数组的运算函数则需要用指定的页面上的二维数组作为输入参数。

**例 4-8** 多维数组参与的数学计算。

**解**：在命令窗口输入：

```
>> A=randn(2,5,2)
A(:,:,1) =
   -1.5175    0.0714    0.4998   -0.5478   -0.0132
    0.0097    0.3165    1.2781    0.2608   -0.5803
A(:,:,2) =
    2.1363   -1.4095    0.3255    0.6204   -0.8960
   -0.2576    1.7701   -1.1190    1.2698    0.1352
>> sum(A)
ans(:,:,1) =
   -1.5078    0.3879    1.7779   -0.2870   -0.5934
ans(:,:,2) =
    1.8787    0.3606   -0.7935    1.8901   -0.7609
>> sin(A)
ans(:,:,1) =
   -0.9986    0.0713    0.4793   -0.5208   -0.0132
    0.0097    0.3113    0.9575    0.2579   -0.5482
ans(:,:,2) =
    0.8443   -0.9870    0.3198    0.5813   -0.7809
   -0.2548    0.9802   -0.8997    0.9550    0.1348
>> eig(A(:,[1 2],1))
ans =
   -1.5179
    0.3169
```

## 4.3 小结

本章介绍了 MATLAB 中创建和操作多维数组的方法。MATLAB 提供了三种创建多维数组的方法和许多获取多维数组属性的函数。MATLAB 中还是采用线性的方式存储多维数组，因此单下标和多下标的索引访问方式之间具有特定的联系。对于多维数组，MATLAB 中提供了类似于二维数组的操作方法，包括对数组形状、维度的重新调整，以及常用的数学计算。



# 第 5 章

# 数据类型概述和数值类型

本章首先介绍 MATLAB 中各种内置数据类型，然后重点讲述 MATLAB 中的数值类型，包括 8 种整数类型和 2 种浮点数据类型。通过本章学习，读者可以对 MATLAB 的内置数据类型有一个简单的了解，并能深入地认识 MATLAB 中各种数值类型之间的区别。

## 5.1 MATLAB 数据类型概述

MATLAB 中有 15 种基本的数据类型，分别是 8 种有符号/无符号整数类型、单精度浮点类型、双精度浮点类型、逻辑数据类型、字符串类型、元胞数组、结构体、函数句柄。在 MATLAB 中，这 15 种基本的数据类型都是按照数组形式存储和操作的。另外，MATLAB 还有两种用于高级交叉编程，用户自定义的面向对象的用户类类型和 Java 类类型。

表 5-1 和图 5-1 给出了 MATLAB 中的 15 种基本数据类型和两种用户自定义类型，并给出了简单的说明。

表 5-1　MATLAB 中的数据类型

| 数据类型 | 示　例 | 说　明 |
|---|---|---|
| int8, uint8,<br>int16, uint16,<br>int32, uint32,<br>int64, uint64 | uint16(7500) | 有符号和无符号的整数类型<br>大部分整数类型占用比浮点类型更少的内存空间<br>除了 int64 和 uint64 类型外的所有整数类型，都可以用在数学计算中 |
| single | single(383.21) | 单精度浮点类型<br>和双精度浮点类型相比，占用内存空间更少，但同时精度和能够表示的数值范围都比双精度浮点类型小 |

续表

| 数据类型 | 示　　例 | 说　　明 |
|---|---|---|
| double | 383.21<br>4+5.2i | 双精度浮点类型<br>是 MATLAB 中默认的数值类型 |
| logical | randn(3,4)>0.5 | 逻辑数据类型<br>1 代表逻辑真，0 代表逻辑假 |
| char | 'welcome!' | 字符串类型 |
| cell array | a{1,1}='hello';<br>a{1,2}=325;<br>a{1,3}=ones(3,2); | 元胞数组类型<br>数组元素可以是不同的数据类型<br>注：存储多个字符串最好用元胞数组类型 |
| structure | a.name='john';<br>a.age=13;<br>a.mat=rand(2,3); | 结构体类型<br>类似于 C 语言，通过结构体中多个成员可以存储多种类型的数据 |
| 函数句柄 | @sin | 函数句柄，相当于一个指针 |

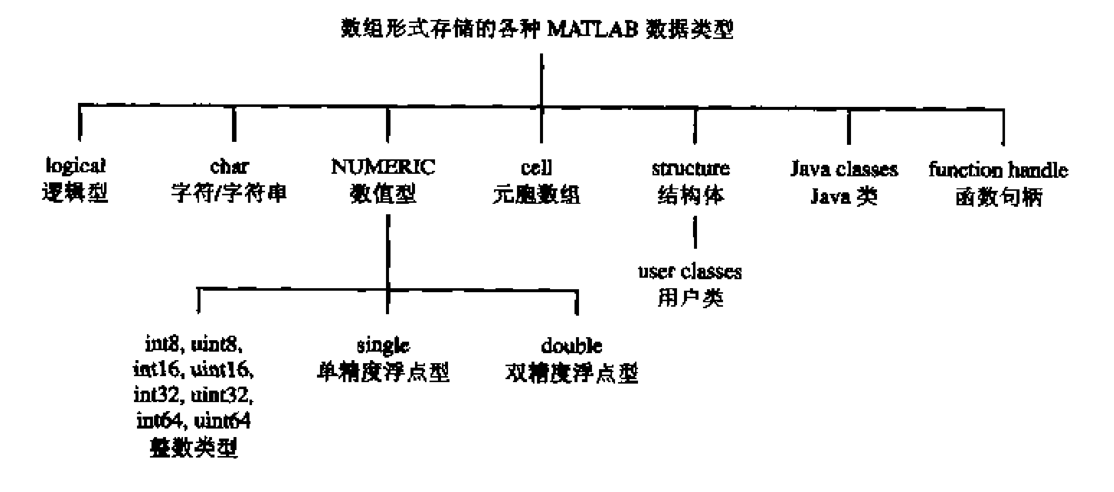



图 5-1　MATLAB 中的数据类型及其关系图

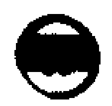
## 5.2　MATLAB 中的数值类型

MATLAB 中的数值类型包括有符号和无符号整数，单精度和双精度浮点数。在默认情况下，MATLAB 按照双精度浮点数类型对所有数值进行存储和操作。

读者也可以通过 MATLAB 提供的函数，指定系统按照多种整数型或单精度浮点型存储和操作单个数字或数组。利用整数型或单精度浮点型的优点在于节省了变量占用的内存空间，当然，这是以降低精度为代价的。

### 5.2.1　整数类型

MATLAB 中提供了 8 种内置的整数类型，表 5-2 列出了它们各自存储占用位数、能表



示的数值范围和转换函数。

表 5-2 MATLAB 中的整数类型

| 整数类型 | 数值范围 | 转换函数 |
| --- | --- | --- |
| 有符号 8 位整数 | $-2^7 \sim 2^7-1$ | int8 |
| 无符号 8 位整数 | $0 \sim 2^8-1$ | uint8 |
| 有符号 16 位整数 | $-2^{15} \sim 2^{15}-1$ | int16 |
| 无符号 16 位整数 | $0 \sim 2^{16}-1$ | uint16 |
| 有符号 32 位整数 | $-2^{31} \sim 2^{31}-1$ | int32 |
| 无符号 32 位整数 | $0 \sim 2^{32}-1$ | uint32 |
| 有符号 64 位整数 | $-2^{63} \sim 2^{63}-1$ | int64 |
| 无符号 64 位整数 | $0 \sim 2^{64}-1$ | uint64 |

不同的整数类型所占用位数不同，因此所能表示的数值范围不同，在实际应用中，应该根据需要的数据范围选择合适的整数类型。有符号的整数类型拿出一位用来表征正负，因此表示的数据范围和相应的无符号整数类型不同。

由于 MATLAB 中数值的默认存储类型是双精度浮点类型，因此，必须通过表 5-2 中列出的转换函数将双精度浮点数值转换成指定的整数类型。在转换过程中，MATLAB 默认将待转换数值转换为最接近的整数，若小数部分正好为 0.5，那么 MATLAB 转换后的结果是绝对值较大的那个整数。另外，应用这些转换函数也可以将其他类型转换成指定的整数类型。

**例 5-1** 通过转换函数创建整数类型。

**解**：在命令窗口输入：

```
>> x=325;y=325.499;z=325.5;
>> xx=int16(x)          %把 double 型的变量 x 强制转换成 int16 型
xx =
    325
>> yy=int32(y)
yy =
          325
>> zz=int32(z)
zz =
          326
>> str='Welcome--!'
str =
Welcome--!
>> int8(str)        %把字符串型的变量强制转换成 int8 型
ans =
   87  101  108   99  111  109  101  126  126   33
>> whos
  Name        Size                    Bytes  Class

  ans         1x10                       10  int8 array
  str         1x10                       20  char array
  x           1x1                         8  double array
```



```
  xx          1x1                    2  int16 array
  y           1x1                    8  double array
  yy          1x1                    4  int32 array
  z           1x1                    8  double array
  zz          1x1                    4  int32 array

Grand total is 26 elements using 64 bytes
```

MATLAB 中还有多种取整函数，可以用不同的策略把浮点小数转换成整数，如表 5-3 所示。

表 5-3　MATLAB 中的取整函数

| 函　数 | 说　明 | 举　例 |
|---|---|---|
| round(*a*) | 向最接近的整数取整<br>小数部分是 0.5 则向绝对值大的方向取整 | round(4.3)结果为 4<br>round(4.5)结果为 5<br>round(−4.5)结果为−5 |
| fix(*a*) | 向 0 取整 | fix(4.3)结果为 4<br>fix(4.5)结果为 4<br>fix(−4.5)结果为−4 |
| floor(*a*) | 向不大于 *a* 的最接近整数取整 | floor(4.3)结果为 4<br>floor(4.5)结果为 4<br>floor(−4.5)结果为−5 |
| ceil(*a*) | 向不小于 *a* 的最接近整数取整 | ceil(4.3)结果为 5<br>ceil(4.5)结果为 5<br>ceil(−4.5)结果为−4 |

整数类型参与的数学运算与 MATLAB 中默认的双精度浮点运算不同。当两种相同的整数类型进行运算时，结果仍然是这种整数类型；当一个整数类型数值与一个双精度浮点类型数值进行数学运算时，计算结果是这种整数类型，取整采用默认的四舍五入（round）方式。需要注意的是，两种不同的整数类型之间不能进行数学运算，除非提前进行强制转换。

**例 5-2**　整数类型数值参与的运算。

**解：**在命令窗口输入：

```
>> x=uint32(367.2)*uint32(20.3)
x =
        7340
>> y=uint32(24.321)*359.63
y =
        8631
>> z=uint32(24.321)*uint16(359.63)
??? Error using ==> mtimes
Integers can only be combined with integers of the same class, or scalar doubles.

>> whos   %显示工作空间的变量
  Name          Size                    Bytes  Class
```

```
  x         1x1                 4  uint32 array
  y         1x1                 4  uint32 array

Grand total is 2 elements using 8 bytes
```

前面的表 5-2 中已经介绍了不同的整数类型能够表示的数值范围不同。数学运算中，运算结果超出相应的整数类型能够表示的范围时，就会出现溢出错误，运算结果被置为该整数类型能够表示的最大值或最小值。

MATLAB 提供了 intwarning 函数可以设置是否显示这种转换或计算过程中出现的溢出、非正常转换的错误，有兴趣的读者可以参考 MATLAB 的联机帮助。

**例 5-3** 整数类型参与的运算及溢出捕获。

**解**：在命令窗口输入：

```
>> int8(300)
ans =
  127
>> int8(100)
ans =
  100
>> int8(100)*3
ans =
  127
>> int8(-300)
ans =
 -128
>> int8(-100)
ans =
 -100
>> int8(-100)*3
ans =
 -128
>> intwarning('on')    %设置显示所有整数型计算、转换中的错误
>> int8(-300)
Warning: Out of range value converted to intmin('int8') or intmax('int8').
ans =
 -128
>> int8(-100)*3
Warning: Out of range value or NaN computed in integer arithmetic.
ans =
 -128
```

## 5.2.2 浮点数类型

MATLAB 中提供了单精度浮点数类型和双精度浮点数类型，它们在存储位宽、各位用处、表示的数值范围、数值精度等方面都不相同，如表 5-4 所示。

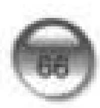




表 5-4 MATLAB 中单精度浮点数和双精度浮点数的比较

| 浮点类型 | 存储位宽 | 各数据位的用处 | 数值范围 | 转换函数 |
|---|---|---|---|---|
| 双精度 | 64 | 0~51 位表示小数部分<br>52~62 位表示指数部分<br>63 位表示符号（0 为正，1 为负） | -1.79769e+308—2.22507e-308<br>2.22507e-308~ 1.79769e+308 | double |
| 单精度 | 32 | 0~22 位表示小数部分<br>23~30 位表示指数部分<br>31 位表示符号（0 为正，1 为负） | -3.40282e+038~ -1.17549e-038<br>1.17549e-038~3.40282e+038 | single |

从表 5-4 可以看出，存储单精度浮点类型所用的位数少，因此内存占用上开支小，但从各数据位的用处来看，单精度浮点数能够表示的数值范围和数值精度都比双精度小。

和创建整数类型数值一样，创建浮点数类型也可以通过转换函数来实现，当然，MATLAB 中默认的数值类型是双精度浮点类型。

**例 5-4** 浮点数转换函数的应用。

**解**：在命令窗口输入：

```
>> x=5.4
x =
     5.4000
>> y=single(x)   %把 double 型的变量强制转换为 single 型并赋值给 y
y =
     5.4000
>> z=uint32(87563);
>> zz=double(z)
zz =
          87563
>> whos
  Name       Size                     Bytes  Class

  x          1x1                          8  double array
  y          1x1                          4  single array
  z          1x1                          4  uint32 array
  zz         1x1                          8  double array

Grand total is 4 elements using 24 bytes
```

双精度浮点数参与运算时，返回值的类型依赖于参与运算中的其他数据类型。双精度浮点数与逻辑型、字符型进行运算时，返回结果为双精度浮点类型；而与整数型进行运算时返回结果为相应的整数类型，与单精度浮点型运算返回单精度浮点型。单精度浮点型与逻辑型、字符型和任何浮点型进行运算时，返回结果都是单精度浮点型。需要注意的是，单精度浮点型不能和整数型进行算术运算。

**例 5-5** 浮点型参与的运算。

**解**：在命令窗口输入：

```
>> x=uint32(240);y=single(32.345);z=12.356;
>> xy=x*y
??? Error using ==> mtimes
Integers can only be combined with integers of the same class, or scalar doubles.
```

```
>> xz=x*z
xz =
        2965
>> yz=y*z
yz =
  399.6548
>> str='welcome';
>> newstr=str-32
newstr =
    87    69    76    67    79    77    69
>> whos
  Name          Size                     Bytes  Class

  newstr        1x7                         56  double array
  str           1x7                         14  char array
  x             1x1                          4  uint32 array
  xz            1x1                          4  uint32 array
  y             1x1                          4  single array
  yz            1x1                          4  single array
  z             1x1                          8  double array

Grand total is 19 elements using 94 bytes
```

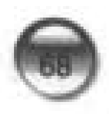
从表 5-4 可以看出，浮点数只占用一定的存储位宽，其中只有有限位分别用来存储指数部分和小数部分。因此，浮点类型能够表示的实际数值是有限的，而且是离散的，任何两个最接近的浮点数之间都有一个很微小的间隙，而所有处在这个间隙中的值都只能用这两个最接近的浮点数中的一个来表示。MATLAB 中提供了 eps 函数，可以获取一个数值和它最接近的浮点数之间的间隙大小。

**例 5-6** 浮点数的精度。

**解**：在命令窗口输入：

```
>> format long
>> eps(5)
ans =
     8.881784197001252e-016
>> eps(50)
ans =
     7.105427357601002e-015
>> eps(single(5))
ans =
   4.7683716e-007
```

例 5-6 中，eps(5)返回和 5 最接近的双精度浮点数与 5 的差值（间隙），这表示在 5~5+eps(5)之间没有计算机可以表示的浮点数存在。因此如果一个计算结果返回双精度浮点数 5，那么它的实际精度不超过 eps(5)。而且从例 5-6 可见，数值 $a$ 越大，eps($a$)也就越大；另外，单精度浮点数之间的间隙比相应的双精度浮点数之间的间隙要大，这是由单精度和双精度浮点数不同的存储方法决定的。





### 5.2.3 复数

复数是对实数的一种扩展，每一个复数包括实部和虚部两部分。MATLAB 中默认用字符 i 或者 j 表示虚部标志。创建复数可以直接输入或者利用 complex 函数。

MATLAB 中还有多种对复数操作的函数，如表 5-5 所示。

表 5-5 MATLAB 中复数相关运算函数

| 函　数 | 说　明 | 函　数 | 说　明 |
|---|---|---|---|
| real(z) | 返回复数 z 的实部 | imag(z) | 返回复数 z 的虚部 |
| abs(z) | 返回复数 z 的模 | angle(z) | 返回复数 z 的辐角 |
| conj(z) | 返回复数 z 的共轭复数 | complex(a,b) | 以 a 为实部，b 为虚部创建复数 |

**例 5-7** 复数的创建和运算。

**解：** 在命令窗口输入：

```
>> a = 2 + 3i

a =

   2.0000 + 3.0000i

>> x = rand(3) * 5;
>> y = rand(3) * -8;
>> z = complex(x, y)  %用 complex 函数创建以 x 为实部，y 为虚部的复数

z =

   4.7506 - 3.5576i   2.4299 - 7.3745i   2.2823 - 3.2456i
   1.1557 - 4.9235i   4.4565 - 5.9057i   0.0925 - 7.4838i
   3.0342 - 6.3355i   3.8105 - 1.4101i   4.1070 - 7.3352i

>> whos
  Name      Size                    Bytes  Class

  a         1x1                        16  double array (complex)
  x         3x3                        72  double array
  y         3x3                        72  double array
  z         3x3                       144  double array (complex)

Grand total is 28 elements using 304 bytes
>> real(z)

ans =

    4.7506    2.4299    2.2823
    1.1557    4.4565    0.0925
    3.0342    3.8105    4.1070

>> conj(z)

ans =
```

```
    4.7506 + 3.5576i    2.4299 + 7.3745i    2.2823 + 3.2456i
    1.1557 + 4.9235i    4.4565 + 5.9057i    0.0925 + 7.4838i
    3.0342 + 6.3355i    3.8105 + 1.4101i    4.1070 + 7.3352i
```

### 5.2.4 无穷量（Inf）和非数值量（NaN）

MATLAB 中用 Inf 和-Inf 分别代表正无穷和负无穷，用 NaN 表示非数值的值。正负无穷的产生一般是由于 0 做了分母或者运算溢出，产生了超出双精度浮点数数值范围的结果；非数值量则是因为 0/0 或者 Inf/Inf 型的非正常运算。需要注意的是，两个 NaN 彼此是不相等的。

除了运算造成这些异常结果外，MATLAB 也提供了专门函数可以创建这两种特别的量，读者可以用 Inf 函数和 NaN 函数创建指定数值类型的无穷量和非数值量，默认是双精度浮点类型。

**例 5-8** 无穷量和非数值量。

**解：**在命令窗口输入：

```
>> x=1/0
Warning: Divide by zero.
x =
   Inf
>> y=exp(1000)
y =
   Inf
>> z=log(0)
Warning: Log of zero.
z =
  -Inf
>> v=Inf
v =
   Inf
>> a=0/0
Warning: Divide by zero.
a =
   NaN
>> b=NaN
b =
   NaN
>> c=NaN('single')          %创建单精度浮点类型的非数值量
c =
   NaN
>> m=7i/0
Warning: Divide by zero.
m =
        NaN +      Infi
>> a==b
ans =
     0
```



## 5.3　数值类型的显示格式

MATLAB 提供了多种数值显示方式，可以通过 format 函数或者 MATLAB 主界面下 File 菜单 Preferences 对话框中修改 Command Window 的设置来使用不同的数值显示方式。默认情况下，MATLAB 使用 5 位定点或浮点型显示格式。

表 5-6 列出 MATLAB 中通过 format 函数提供的几种数值显示格式，并举例加以说明。

表 5-6　format 函数设置数值显示格式

| 函数形式 | 说　明 | 举　例 |
|---|---|---|
| format short | 5 位定点显示格式（默认） | 3.1416 |
| format short e | 5 位带指数浮点显示格式 | 3.1416e+000 |
| format long | 15 位定点浮点显示格式（单精度浮点数用 7 位） | 3.14159265358979 |
| format long e | 15 位带指数浮点显示格式（单精度浮点数用 7 位） | 3.141592653589793e+000 |
| format bank | 小数点后保留两位的银行格式 | 3.14 |
| format rat | 分数有理近似格式 | 355/113 |

format 函数和 Preferences 对话框都只修改数值的显示格式，而 MATLAB 中数值运算都不受影响，按照双精度浮点运算进行。

在 MATLAB 编程中，还常需要临时改变数值显示格式，这可以通过 get 和 set 函数来实现，下面举例加以说明。

**例 5-9**　通过 get 和 set 临时改变数值显示格式。

**解**：在命令窗口输入：

```
>> origFormat = get(0, 'format')   %获取当前的数值显示格式并保存在变量 origFormat 中
origFormat =
short
>> format('rational')
>> rat_pi=pi
rat_pi =
      355/113
>> rat_e=exp(1)
rat_e =
     1457/536
>> set(0,'format', origFormat)      %将数值显示格式重新设置为之前保存在变量 origFormat 中的值
>> get(0, 'format')
ans =
short
```

## 5.4　MATLAB 中确定数值类型的函数

除了前面各节中介绍的数值相关函数外，MATLAB 中还有很多用于确定数值类型的函数，如表 5-7 所示。

表 5-7　MATLAB 中确定数值类型的函数

| 函　　数 | 用　　法 | 说　　明 |
|---|---|---|
| class | class(*A*) | 返回变量 *A* 的类型名称 |
| isa | isa(*A*,'class_name') | 确定变量 *A* 是否为 class_name 表示的数据类型 |
| isnumeric | isnumeric(*A*) | 确定 *A* 是否为数值类型 |
| isinteger | isinteger(*A*) | 确定 *A* 是否为整数类型 |
| isfloat | isfloat(*A*) | 确定 *A* 是否为浮点类型 |
| isreal | isreal(*A*) | 确定 *A* 是否为实数 |
| isnan | isnan(*A*) | 确定 *A* 是否为非数值量 |
| isinf | isinf(*A*) | 确定 *A* 是否为无穷量 |
| isfinite | isfinite(*A*) | 确定 *A* 是否为有限数值 |

## 5.5　小结

本章开始概述了 MATLAB 中的基本数据类型和用户数据类型，然后详细介绍了整数型和浮点型共 10 种数值类型的存储方式、数值范围和应用优缺点等，对各种数值类型参与的运算进行了详细的介绍，尤其对其中运算可能出现的溢出、精度问题进行了分析，最后还介绍了 MATLAB 中控制显示数值显示格式的 format 函数和多种确定数值类型的内联函数。



# 第 6 章

# 结构体和元胞数组

本章介绍 MATLAB 中两种复杂的数据类型：结构体（Structure）和元胞数组（Cell Array）。这两种类型都可以存储多组不同类型的数据，在程序设计中应用广泛。本章主要讲解这两种数据类型的创建、其内部数据的索引寻址，以及其他的操作函数。

## 6.1 结构体

MATLAB 中的结构体和 C 语言中类似，一个结构体可以通过字段存储多个不同类型的数据，因此，结构体相当于一个数据容器，把多个相关联的不同类型的数据封装在一个结构体对象当中。

如图 6-1 所示，结构体 patient 中有三个字段：姓名字段 name 中存储了一个字符串类型的数据；账单字段 billing 中存储了一个浮点数值；test 字段中存储了一个二维数组。一个结构体中可以具有多个字段，它们又可以存储不同类型的数据，通过这种方式，就可以把多个不同类型的数据组织在一个结构体对象中。

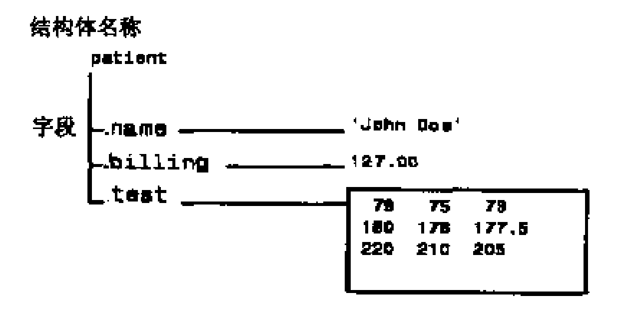



图 6-1 结构体 patient 的图示

MATLAB 中，一个结构体对象就是一个 1*1 的结构体数组，因此，可以创建具有多个结构体对象的二维或多维结构体数组。

### 6.1.1 结构体的创建

创建结构体对象有两种方法：直接采用赋值语句给结构体的字段赋值；通过结构体创建函数 struct 来创建。

**1. 采用赋值语句创建结构体**

在对结构体的字段进行赋值时，赋值表达式的左边代表了结构体的字段变量名，用"结构体名称.字段名称"的形式书写，对一个结构体多个字段可以赋值。

**例 6-1** 通过对字段赋值创建结构体。

**解：** 在命令窗口输入：

```
>> patient.name = 'John Doe';
>> patient.billing = 127.00;
>> patient.test = [79 75 73; 180 178 177.5; 220 210 205];
>> patient
patient =
      name: 'John Doe'
   billing: 127
      test: [3x3 double]
>> whos
  Name          Size                    Bytes  Class

  patient       1x1                       468  struct array

Grand total is 21 elements using 468 bytes
```

在例 6-1 中，通过对三个字段（name，billing，test）赋值，创建了一个具有三个字段的结构体对象 patient。用 whos 显示发现 patient 是一个 1*1 的结构体数组。

通过圆括号索引指派，用字段赋值的方法还可以创建任何尺寸的结构体数组。

需要注意的是，同一个结构体数组中的所有结构体对象具有相同的字段，没有明确赋值的字段，MATLAB 默认赋值为空数组。

**例 6-2** 通过圆括号索引指派，用字段赋值的方法创建结构体数组。

**解：** 在命令窗口输入：

```
>> patient(1).name = 'John Doe';
>> patient(1).billing = 127.00;
>> patient(1).test = [79 75 73; 180 178 177.5; 220 210 205];
>> whos
  Name          Size                    Bytes  Class

  patient       1x1                       468  struct array

Grand total is 21 elements using 468 bytes

>> patient(2).name = 'Ann Lane';
>> patient(2).billing = 28.50;
```



```
>> patient(2).test = [68 70 68; 118 118 119; 172 170 169];
>> whos
  Name          Size                    Bytes  Class

  patient       1x2                       744  struct array

Grand total is 42 elements using 744 bytes

>> patient
patient =
1x2 struct array with fields:
    name
    billing
    test
>> patient(3).name = 'Alan Johnson'
patient =
1x3 struct array with fields:
    name
    billing
    test
>> patient(3).billing
ans =
     []
>> patient(3).test
ans =
     []
```

### 2. 采用 struct 函数创建结构体

除了通过字段赋值，还可以用 struct 函数创建结构体。

struct 函数的语法形式为 strArray = struct('field1',val1,'field2',val2,…)，其功能为创建结构体对象 strArray，并将其第 *n* 个字段 fieldn 赋值为 valn。

这一语句中 valn 可以是具有相同尺寸的元胞数组，这样则会创建和元胞数组具有相同尺寸的结构体数组 strArray，并将各个结构体元素的字段 fieldn 赋值为 valn 元胞数组中对应位置上的取值。

**例 6-3**　利用 struct 函数创建结构体数组。

**解**：在命令窗口输入：

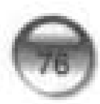
```
>> weather(2)= struct('total','sunny','temp',18,'rainfall',0.0)
weather =
1x2 struct array with fields:
    total
    temp
    rainfall
>> weather(1)    %结构体数组的第一个元素没有赋值，因此所有字段赋值为空数组
ans =
       total: []
        temp: []
    rainfall: []
>> weather(2)
```

```
ans =
      total: 'sunny'
       temp: 18
   rainfall: 0
>> weaArray=repmat(struct('total','sunny','temp',18,'rainfall',0.0),1,3)
weaArray =
1x3 struct array with fields:
    total
    temp
    rainfall
>> weaArray(1)        %1*3 的结构体数组的三个结构体元素相同
ans =
      total: 'sunny'
       temp: 18
   rainfall: 0
>> weaArray(2)
ans =
      total: 'sunny'
       temp: 18
   rainfall: 0
>> weaArray(3)
ans =
      total: 'sunny'
       temp: 18
   rainfall: 0
>>  newArray=struct('total',{'sunny','rainy'},'temp',{18,10},'rainfall',
{0.0,1.5})
newArray =
1x2 struct array with fields:
    total
    temp
    rainfall
>> newArray(1)
ans =
      total: 'sunny'
       temp: 18
   rainfall: 0
>> newArray(2)
ans =
      total: 'rainy'
       temp: 10
   rainfall: 1.5000
```

最后补充说明一下，结构体在内存中的存储并不需要连续的内存块，只是结构体的每一个字段存储在一块连续内存中。

### 6.1.2 获取结构体内部数据

对于结构体数组，也可以通过括号、下标索引来访问其内部的数据，需要注意的是结构体名和字段之间用点（.）连接。







对于如图 6-2 所示的结构体，例 6-4 给出了访问其内部数据的多种方法。

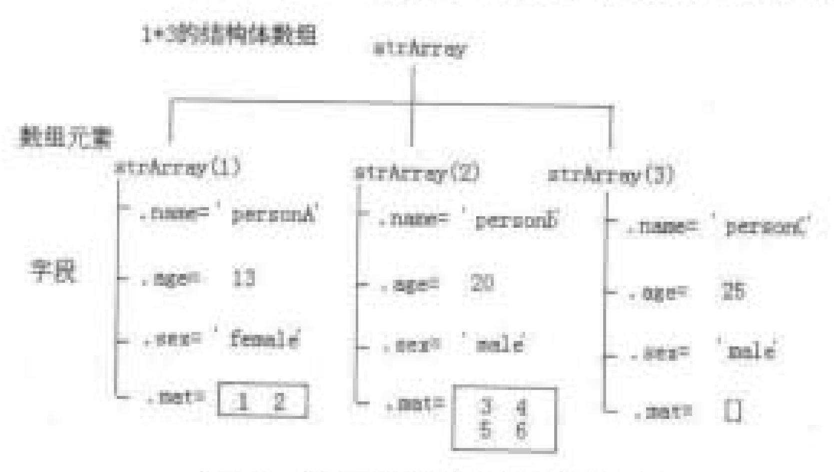



图 6-2　例 6-4 使用的结构体数组的图示

**例 6-4**　结构体内部数据的获取。

**解**：在命令窗口输入：

```
>> strArray=struct('name',{'personA','personB','personC'}...
,'age',{13,20,25},'sex',{'female','male','male'},...
'mat',{[1 2],[3 4;5 6],[]})      %创建结构体数组（如图 6-2 所示）
strArray =
1x3 struct array with fields:
    name
    age
    sex
    mat
>> newArray=strArray(1:2)    %通过冒号和括号裁剪产生结构体数组的子数组
newArray =
1x2 struct array with fields:
    name
    age
    sex
    mat
>> strArray(1)              %通过下标索引访问结构体数组中的第一个结构体元素
ans =
    name: 'personA'
     age: 13
     sex: 'female'
     mat: [1 2]
>> strArray(2).name         %访问第二个结构体元素的 name 字段值
ans =
personB
>> strArray(2).mat(2,1)     %访问第二个结构体元素的 mat 字段数组的第 2 行第 1 列元素
ans =
     5
>> ageArray=strArray.age    %多个结构体对象的同一字段值不能赋值给一个数组
??? Illegal right hand side in assignment. Too many elements.

>> ageArray={strArray.age}  %多个结构体对象的同一字段值只能赋值给元胞数组
ageArray =
```

```
    {13}    {20}    {25}
>> whos ageArray             %ageArray是一个1*3的元胞数组
  Name          Size                    Bytes  Class

  ageArray      1x3                       204  cell array

Grand total is 6 elements using 204 bytes
```

### 6.1.3 结构体数组操作函数

对结构体数组的常见操作，包括获取其尺寸信息和增删字段。

和一般的数组一样，size 函数可以获取结构体数组的尺寸，也就是数组中包含多少行、多少列个结构体对象。当然，如果 size 函数的输入参数是结构体字段，则返回的是该字段的尺寸信息。

向结构体中增加新的字段，只需要添加新的字段赋值语句即可。没有被明确赋值的结构体中，添加的新字段被初始化为空数组。

从结构体中删除字段，需要用到 rmfield 函数，其语法格式为 newstrArray=rmfield (strArray,'fieldname')，表示从结构体数组 strArray 中删除 fieldname 代表的字段，并把结果赋值给新的结构体数组 newstrArray。

下面还以图 6-2 所示的结构体数组 strArray 为例，对结构体各种操作方法给出实例。

**例 6-5** 结构体数组的操作。

**解**：在命令窗口输入：

```
>> whos
  Name          Size                    Bytes  Class

  strArray      1x3                      1118  struct array

Grand total is 56 elements using 1118 bytes

>> size(strArray)              %获取结构体数组的尺寸
ans =
     1     3
>> size(strArray(2).mat)   %获取单个结构体对象的指定字段的尺寸
ans =
     2     2
>> strArray(2).ID=357;      %向结构体数组中的每个元素增加新字段ID，并对第二个元素
                             的该字段赋值为357
>> strArray(1)
ans =
      name: 'personA'
       age: 13
       sex: 'female'
       mat: [1 2]
        ID: []
>> strArray(2)
ans =
      name: 'personB'
```



```
     age: 20
     sex: 'male'
     mat: [2x2 double]
      ID: 357
>> strArray=rmfield(strArray,'mat')     %删除所有结构体元素中的 mat 字段
strArray =
1x3 struct array with fields:
    name
    age
    sex
    ID
```

### 6.1.4 结构体嵌套

前面已经讲过，结构体可以有多个字段，可以存储多种类型的数据，结构体的字段中也能存储结构体类型的数据，这就是结构体嵌套。

嵌套的结构体创建方法和 6.1.1 讲过的方法类似，只不过该结构体某个字段值为结构体类型的数据。

对嵌套结构体的内部数据的访问也和 6.1.2 所讲相似，嵌套在内层的结构体作为外层结构体的字段，因此在内外层结构体名称之间用点号（.）连接，所以数据访问的表达式会更加复杂一些。

例 6-6 给出一个结构体嵌套的实例，并对其内部数据的访问也给出示范，其结构体图示如图 6-3 所示。

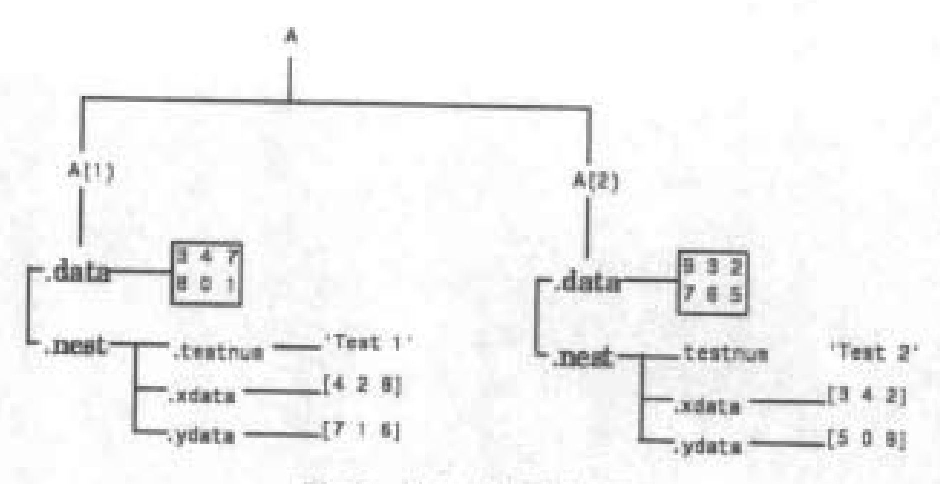



图 6-3 例 6-6 中的嵌套结构体

**例 6-6** 结构体嵌套。

**解**：在命令窗口输入：

```
>> A = struct('data', [3 4 7; 8 0 1], 'nest',...
   struct('testnum', 'Test 1', 'xdata', [4 2 8],...
   'ydata', [7 1 6]));             %创建嵌套结构体
>> A(2).data = [9 3 2; 7 6 5];  %向嵌套结构体数组中增加一个元素并对各字段赋值
>> A(2).nest=struct('testnum','Test 2','xdata',[3 4 2],'ydata',[5 0 9]);
>> A(1).data    %访问嵌套结构体数组中第一个元素的 data 字段
```

```
ans =
     3     4     7
     8     0     1
>> A(1).nest       %访问嵌套结构体数组中第一个元素的 nest 字段，返回结果是一个结构体
ans =
    testnum: 'Test 1'
      xdata: [4 2 8]
      ydata: [7 1 6]
>> A(1).nest.xdata         %访问嵌套结构体的内部数据
ans =
     4     2     8
>> A(1).nest.ydata(2)
ans =
     1
```

### 6.1.5 动态字段

访问结构体中指定字段的数据，可以用 6.1.2 讲过的方法。但当字段也是变量的时候，访问结构体内部的数据就需要用到动态字段，这经常用在 MATLAB 程序设计中。动态字段访问的方法和指定字段类似，只是在动态字段变量名两边添加圆括号。

例 6-7 中的函数 getmean 用于计算结构体 strArray 中 fieldname 字段的数据均值。实际使用中，可以改变 strArray 和 fieldname 变量的输入值，从而计算不同结构体不同字段的均值。

**例 6-7** 动态字段的访问。

**解**：在命令窗口输入：

```
函数代码：
function y=getmean(strArray,fieldname)
y=mean(strArray.(fieldname));
命令窗口中函数测试代码：
>> A=struct('xdata',[3 1 2],'ydata',[6 7 8 9],'zdata',repmat(100,1,4))
A =
    xdata: [3 1 2]
    ydata: [6 7 8 9]
    zdata: [100 100 100 100]
>> getmean(A,'xdata')
ans =
     2
>> getmean(A,'ydata')
ans =
    7.5000
>> getmean(A,'zdata')
ans =
   100
```

### 6.1.6 结构体函数

MATLAB 中提供了大量与结构体相关函数，如表 6-1 所示。

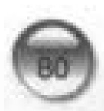




表 6-1　MATLAB 中的结构体函数

| 函　数 | 功　能 |
| --- | --- |
| deal | 将输入变量对应赋值给输出变量 |
| fieldnames | 返回结构体的所有字段，返回值是以字符串为元素的元胞数组类型 |
| isfield | 测试某字符串是否为指定结构体的字段 |
| isstruct | 测试某变量是否为结构体类型 |
| rmfield | 删除结构体字段 |
| struct | 创建结构体 |
| struct2cell | 把结构体转换成元胞数组 |

**例 6-8**　结构体函数的使用。

**解：** 在命令窗口输入：

```
>> A=struct('name','A','grade',2,'score',81) %创建结构体
A =
     name: 'A'
    grade: 2
    score: 81
>> fieldnames(A)  %返回结构体的所有字段
ans =
    'name'
    'grade'
    'score'
>> isstruct(A)  %测试 A 是否是结构体类型
ans =
     1
>> isfield(A,'grade')     %测试 grade 是否是指定结构体的字段
ans =
     1
>> A=rmfield(A,'grade')  %删除结构体字段 grade
A =
     name: 'A'
    score: 81
>> struct2cell(A)          %把结构体转换成元胞数组
ans =
    'A'
    [81]
```

## 6.2　元胞数组

和结构体类型类似，元胞数组也可以存储不同类型、不同尺寸的数据。元胞数组是对常规的数值数组的扩展，其每一个元素称为一个元胞，每一个元胞中可以存储任意类型、任意尺寸的数据。

图 6-4 就是一个 2*3 的元胞数组的图示，其每一个元胞中可以存储二维数组、结构体甚至另一个元胞数组。

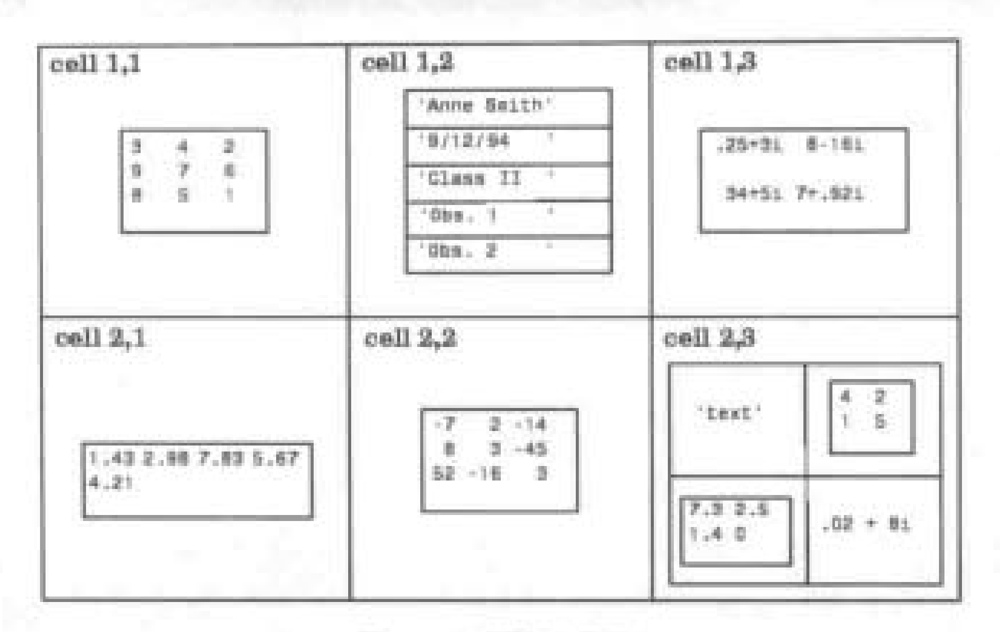



图 6-4　元胞数组的图示

### 6.2.1　元胞数组的创建

和结构体的创建类似，元胞数组的创建也有两种方法：直接对各个元胞赋值，或者用 cell 函数创建。

#### 1．直接赋值创建元胞数组

元胞数组的各类操作中，经常要用到花括号（{}）。在赋值语句中，花括号有两种使用方法：

（1）花括号用在下标索引上，则出现在赋值语句等号左侧，那么右侧只写索引表示的位置上元胞内的数据；

（2）若左侧下标索引用圆括号括起来，则右侧的元胞数据要用花括号括起来。

#### 2．采用 cell 函数创建元胞数组

cell 函数创建元胞数组的语法格式为：

arrayName=cell(*m*,*n*)

其意义是创建包含 *m* 行 *n* 列个元胞的元胞数组 arrayName。

这一语句只用于元胞数组的预声明，之后还需要对每一个元胞内的数据进行初始化赋值，这就和第一种方法类似了。

**例 6-9**　创建元胞数组。

**解**：在命令窗口输入：

```
>> A(1,1) = {[1 4 3; 0 5 8; 7 2 9]};
>> A(1,2) = {'Anne Smith'};
>> A(2,1) = {3+7i};
>> A(2,2) = {-pi:pi/10:pi};
```



```
>> A
A =
        [3x3 double]    'Anne Smith'
   [3.0000 + 7.0000i]    [1x21 double]
>> B(3,3)={'Hello'};
>> B
B =
     []      []         []
     []      []         []
     []      []    'Hello'
>> C=cell(3,4);
>> C{2,3}=rand(2);
>> C
C =
     []      []              []      []
     []      []    [2x2 double]      []
     []      []              []      []
>> whos
  Name      Size                    Bytes  Class

  A         2x2                       516  cell array
  B         3x3                       102  cell array
  C         3x4                       136  cell array

Grand total is 75 elements using 754 bytes
```

通过例 6-9 可见，创建元胞数组时，没有明确赋值的元胞被 MATLAB 默认赋值为空数组。

圆括号括起下标只能够定位元胞数组中的每一个单元，而这些单元是元胞类型的，要访问元胞中存储的数据，则要用花括号括起下标来访问。

## 6.2.2　元胞数组的显示

MATLAB 中默认用紧密格式显示元胞数组。这种格式下，绝大部分类型的元胞只显示其数据类型和尺寸，而不显示具体的的数据内容，如例 6-9 所示。

MATLAB 提供了 celldisp 函数，可以用来逐个显示元胞的具体数据内容。

cellplot 函数则可以绘制出图像，显示元胞数组的结构。

**例 6-10**　元胞数组的显示。

**解**：在命令窗口输入：

```
>> C{1,1}={[1 2;3 4]};
>> C{1,2}={'string'};
>> C{2,1}={struct('name','liuliu','age',20,'sex','male')};
>> C{2,2}={struct2cell(struct('fieldA','a','fieldB','b','fieldC','c'))};
>> C
C =
    [2x2 double]    'string'
    [1x1 struct]    {3x1 cell}
>> celldisp(C)
C{1,1} =
```

```
     1     2
     3     4
C{2,1} =
    name: 'liuliu'
     age: 20
     sex: 'male'
C{1,2} =
string
C{2,2}{1} =
a
C{2,2}{2} =
b
C{2,2}{3} =
c
>> cellplot(C)        %结果如图 6-5 所示
```

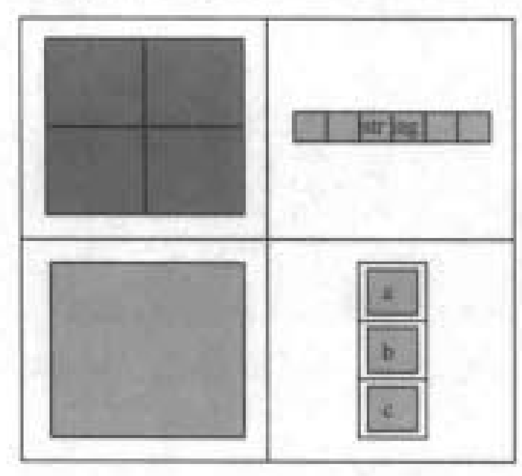



图 6-5　例 6-10 中 cellplot(C)的结果

### 6.2.3　元胞数组的操作

本节介绍元胞数组的数据访问、元胞删除和数组重定义尺寸等操作。

元胞数组数据访问的方法有两种：

（1）用花括号括起的下标索引能够直接访问元胞单元内的数据；

（2）用圆括号括起来的下标索引只能定位元胞单元位置，访问返回结果仍然是一个元胞类型的数组（特殊情况下为 1*1 的元胞数组），通常用于从较大的元胞数组中裁剪产生数组子集。

**例 6-11**　元胞数组的数据访问。

**解：**在命令窗口输入：

```
>> N{1,1} = [1 2; 4 5];
>> N{1,2} = 'Name';
>> N{2,1} = 2-4i;
>> N{2,2} = 7;
>> c = N{1,2}
c =
Name
```



```
>> d = N{1,1}(2,2)
d =
     5
>> cc=N(1,2)
cc =
    'Name'
>> whos c cc
  Name      Size                    Bytes  Class

  c         1x4                         8  char array
  cc        1x1                        68  cell array

Grand total is 9 elements using 76 bytes
>> dd=N(:)
dd =
    [2x2 double]
    [2.0000 - 4.0000i]
    'Name'
    [             7]
>> whos dd
  Name      Size                    Bytes  Class

  dd        4x1                       304  cell array

Grand total is 14 elements using 304 bytes
```

需要删除元胞数组中某些单元时，只需要用圆括号括起目标单元的下标，并赋值为空数组即可。

**reshape** 函数可以改变元胞数组的形状，在改变形状时还是按照行-列-页-…的优先顺序。

**例 6-12**　删除元胞和改变元胞数组形状。

**解：** 在命令窗口输入：

```
>> C={'cell(1,1)','cel(1,2)','cell(1,3)';…
'cell(2,1)','cell(2,2)','cell(2,3)'}
C =
    'cell(1,1)'    'cel(1,2)'     'cell(1,3)'
    'cell(2,1)'    'cell(2,2)'    'cell(2,3)'
>> cellplot(C)        %结果如图 6-6(a)所示
>> D=reshape(C,3,2)
D =
    'cell(1,1)'    'cell(2,2)'
    'cell(2,1)'    'cell(1,3)'
    'cel(1,2)'     'cell(2,3)'
>> C(:,2)=[]
C =
    'cell(1,1)'    'cell(1,3)'
    'cell(2,1)'    'cell(2,3)'
>> cellplot(C)        %结果如图 6-6(b)所示
```

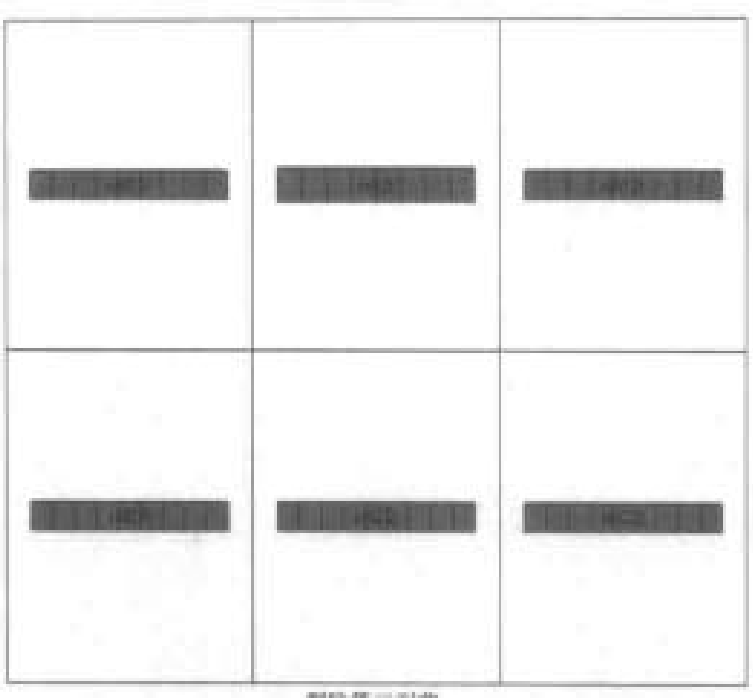



图 6-6（a） 例 6-12 cellplot(C)结果一

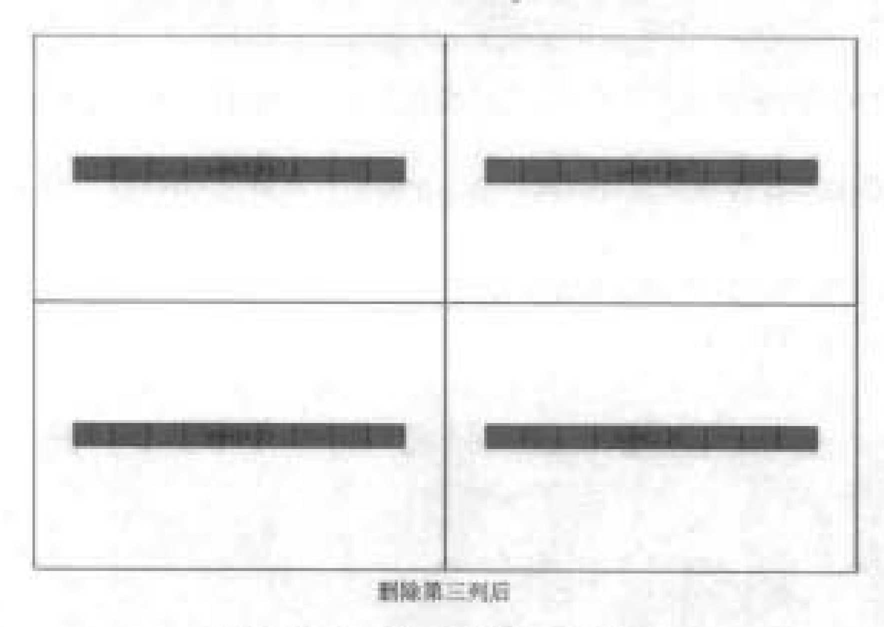



图 6-6（b） 例 6-12 cellplot(C)结果二

### 6.2.4 嵌套元胞数组

和结构体类型类似，元胞数组也可以互相嵌套，就是一个元胞单元中存储了元胞数组类型的数据。一个简单的例子如图 6-7 所示。



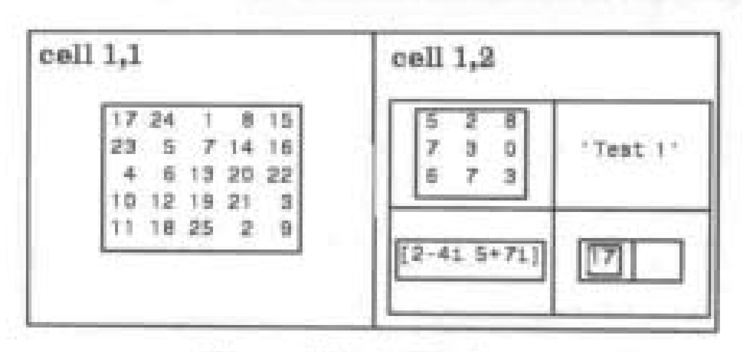



图 6-7　嵌套元胞数组的图示

嵌套元胞数组的创建和各种操作都与一般的元胞数组类似，只是各种语句要复杂一些而已。例 6-13 创建图 6-7 所示的元胞数组，并示例嵌套元胞数组内部数据的访问方法。

**例 6-13**　嵌套元胞数组的创建和操作。

**解**：在命令窗口输入：

```
>> A{1,1} = {magic(5)};
>>A{1,2}{1,1} = {[5 2 8; 7 3 0; 6 7 3]};    %A{1,2}是一个嵌套在内层的元胞数组
>> A{1,2}{1,2} = {'Test 1'};
>> A{1,2}{2,1} = {[2-4i 5+7i]};
>> A{1,2}{2,2} = {cell(1, 2)}
A =
    [5x5 double]    {2x2 cell}
>> A{1,2}{2,2}{1} = {17};
>> celldisp(A)
A{1} =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
A{2}{1,1} =
     5     2     8
     7     3     0
     6     7     3
A{2}{2,1} =
   2.0000 - 4.0000i   5.0000 + 7.0000i
A{2}{1,2} =
Test 1
A{2}{2,2}{1} =
    17
A{2}{2,2}{2} =
     []
>> A{1,1}(3,4)        %访问元胞数组 A 第 1 行第 1 列的元胞中的第 3 行第 4 列的数据
ans =
    20
>> A{1,2}{2,1}(2)    %访问元胞数组 A 第 1 行第 2 列的第 2 行第 1 列的元胞中的第 2 个数据
ans =
   5.0000 + 7.0000i
```

### 6.2.5 元胞数组函数

MATLAB 中的元胞数组函数如表 6-2 所示。

表 6-2 元胞函数

| 函　数 | 说　明 |
| --- | --- |
| deal | 将输入变量对应赋值给输出变量 |
| cell | 创建指定尺寸的元胞数组 |
| celldisp | 逐个显示元胞单元的数据 |
| cellplot | 用图形方式显示元胞数组结构 |
| cell2struct | 把元胞数组转换成结构体 |
| num2cell | 把数值数组转换成元胞数组 |
| iscell | 测试某个对象是否为元胞类型 |

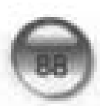
**例 6-14** 元胞函数的应用。

**解：**在命令窗口输入：

```
>> A=[1 16;2 20;3 25]
A =
     1    16
     2    20
     3    25
>> B=num2cell(A)
B =
   [1]    [16]
   [2]    [20]
   [3]    [25]
>> iscell(B)
ans =
     1
>> C=cell2struct(B,{'ID','age'},2)  %按 B 的列方向，把元胞数组转换成具有指定字段的结构体数组
C =
3x1 struct array with fields:
    ID
    age
>> C(1)
ans =
     ID: 1
    age: 16
```

## 6.3 小结

本章讲解了 MATLAB 中两种比较复杂的数据类型——结构体和元胞数组，重点介绍了这两种数据类型的创建和操作方法。通过本章的学习，读者能够理解这两种数据类型的特点和使用方法，掌握对结构体和元胞数组内部数据的访问方法。



# 第 7 章 字符串

本章介绍 MATLAB 中的字符和字符串。字符串类型经常用于代码标记、错误和警告输出、图形图像的标注、本地文件操作等方面，因此也是极重要的数据类型。本章主要讲解 MATLAB 中字符串类型的创建、比较、查找替代和与数值数组之间的转换等操作。

## 7.1 创建字符串

在 MATLAB 中，单个字符是按照 Unicode 编码存储的，每一个字符占两个字节。MATLAB 内部用字符的编码数值对字符/字符串进行运算，因此，字符串本质上是一个数字串，不过其输出到屏幕时则按照字符格式显示。

以字符为元素的数组称为字符数组。在 MATLAB 中，字符数组可以分为单行字符数组和多行字符数组。

### 7.1.1 单行字符串创建

创建单行的字符串很方便，只需要把字符内容用单引号（'）括起来即可，还可以用方括号（[]）连接多个字符串组成较长的字符串，或者用 strcat 函数将多个字符串横向连接成更长的字符串。

需要注意的是，用方括号连接的时候，字符串中所有的空格都会保留下来，而用 strcat 函数连接字符串的时候，被连接的每一个字符串最右边的所有空格将被裁切。单行的字符串的创建，如例 7-1 所示，比较简单。

**例 7-1** 字符串的创建。

**解**：在命令窗口输入：

```
>> a='  a';b='b  b';c='c   ';      %用单引号创建字符串，都由四个字符组成
>> length(a),length(b),length(c)   %获取字符串长度
ans =
     4
ans =
     4
ans =
     4
>> ABC=[a b c]      %用方括号连接三个字符串，空格不被裁切
ABC =
   ab  bc
>> length(ABC)
ans =
    12
>> sAC=strcat(a,c)  %用 strcat 函数连接字符串，每个字符串最右边的空格被裁切
sAC =
   ac
>> length(sAC)
ans =
     5
>> AC=[a,c]
AC =
   ac
>> length(AC)
ans =
     8
>> whos      %显示工作区内所有变量，每一个字符占用两个字节
  Name      Size                    Bytes  Class

  ABC       1x12                       24  char array
  AC        1x8                        16  char array
  a         1x4                         8  char array
  ans       1x1                         8  double array
  b         1x4                         8  char array
  c         1x4                         8  char array
  sAC       1x5                        10  char array

Grand total is 38 elements using 82 bytes
```

### 7.1.2 多行字符串创建

MATLAB 中经常要用到多行字符串，二维字符数组是最简单的多行字符串，下面以二维字符数组的创建来讲述多行字符串创建。

二维字符数组相当于把多个字符串纵向连接起来。这种连接也可以用方括号，只是在各行字符串之间要用分号（;）分隔，而不再是横向连接时用的逗号或者空格了。

另外，MATLAB 还提供了 strvcat 函数和 char 函数用于纵向连接多个字符串。用方括号纵向连接的每一行字符串，要求长度相等，否则必须换用函数方法连接。

strvcat 函数连接多行字符串时，不要求每行长度相等，MATLAB 会自动把非最长行的字符串最右边补空格，使所有行和最长的字符串具有相同长度。



char 函数和 strvcat 函数类似，不过 strvcat 函数会自动忽略空字符串，而 char 函数连接时，会把空字符串也用空格填满，然后连接。

**例 7-2** 创建二维字符数组。

**解：** 在命令窗口输入：

```
>> a='a';
>> b='bb';
>> c='';      %创建空字符串
>> [a;b]      %不同长度的字符串不能用方括号纵向连接
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.

>> [a,' ';b]      %把第一行字符串用空格扩展成和第二行等长
ans =
a
bb
>> sAB=strvcat(a,b) %strvcat 连接，不要求每行等长
sAB =
a
bb
>> size(sAB)
ans =
     2     2
>> sABC=strvcat(a,b,c)  %strvcat 连接，空字符串被忽略
sABC =
a
bb
>> size(sABC)
ans =
     2     2
>> cABC=char(a,b,c) %char 连接，空字符串也被空格填满
cABC =
a
bb

>> size(cABC)
ans =
     3     2
```

## 7.2 字符串操作

MATLAB 中，对字符串常用的操作有：字符串的比较、字符串的替换和查找、字符串与数值数组之间的转换等。

### 7.2.1 字符串比较

字符串的比较操作包括：

（1）比较两个字符串是否完全的或者部分的相同；

（2）比较两个字符串中逐个字符是否相同；

（3）检查字符串中的单个字符是否为字母、数字等。

MATLAB 中比较两个字符串是否相同的函数有 strcmp，strncmp，strcmpi 三个。

（1）strcmp 可以比较两个字符串是否完全相同。若是，则返回逻辑真；否则返回逻辑假。

（2）strncmp 可以比较两个字符串的前 *n* 个字符是否相同。若是，返回真；否则返回假。

（3）strcmpi 可以比较两个字符串是否完全相同。

strcmpi 与 strcmp 类似，不过比较的时候忽略字母大小写的差别。

strncmpi 与 strncmp 类似，区别同样是 strncmpi 比较两个字符串前 *n* 个字符时，忽略字母大小写的差别。

**例 7-3** 字符串的比较。

**解**：在命令窗口输入：

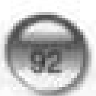
```
>> a='abcde';
>> b='abcgh';
>> c='abcDE';
>> d='aBCde';
>> strcmp(a,b)
ans =
     0
>> strncmp(a,b,3)
ans =
     1
>> strcmp(a,c)         %strcmp 比较，有大小写差别
ans =
     0
>> strcmpi(a,c)        %ctrcmpi 比较时，忽略大小写差别
ans =
     1
>> strncmp(b,d,3)
ans =
     0
>> strncmpi(b,d,3)
ans =
     1
```

两个字符串还可以逐个字符的比较，MATLAB 中用关系运算符等于（==）实现这一比较。需要注意的是，待比较的两个字符串必须长度相等，或者其中之一为单个字符。

**例 7-4** 两个字符串逐个字符的比较。

**解**：在命令窗口输入：

```
>> a='aa';
>> aa='aa  ';
>> b='abcd';
>> c='c';
>> a==b %不等长的字符串不能用==比较
??? Error using ==> eq
Matrix dimensions must agree.
```





```
>> aa==b
ans =
     1     0     0     0
>> c==b %单个字符和字符串之间可以用==比较
ans =
     0     0     1     0
```

在实际处理字符串时，经常还会遇到确定字符串的某个字符属于英文字母、数字还是空格等格式字符的问题。

MATLAB 中的几个字符归属测试函数可以用来检测字符串中的字符。这些函数是：

（1）isletter 可以检测字符串中的每一个字符是否属于英文字母；

（2）isspace 可以检测字符串中的字符是否属于格式字符（空格、制表符、回车和换行等）；

（3）isstrprop 函数可以逐个检测字符是否属于指定的范围。

例 7-5 给出前两个函数的应用示范，有关 isstrprop 函数的使用请读者参考 MATLAB 的联机帮助。

**例 7-5** 字符归属测试函数。

**解**：在命令窗口输入：

```
>> a='Building 17#';
>> isletter(a)
ans =
     1     1     1     1     1     1     1     1     0     0     0     0
>> isspace(a)
ans =
     0     0     0     0     0     0     0     0     1     0     0     0
```

## 7.2.2 字符串的替换和查找

MATLAB 中用 strrep 函数进行字符串的替换。strrep 的语法格式为：

```
strrep(str1,str2,str3)
```

它把 str1 中所有的 str2 子串用 str3 来替换。

需要注意，strrep 对字母的大小写敏感，只能替换 str1 中与 str2 完全一致的子串。

**例 7-6** 字符串的替换。

**解**：在命令窗口输入：

```
>> str='Error Msg';
>> newstr=strrep(str,'Msg','Mail')
newstr =
Error Mail
>> newstrr=strrep(str,'msg','Mail')     %strrep 对大小写敏感，替换没有进行
newstrr =
Error Msg
```

MATLAB 中有丰富的字符串查找函数，如表 7-1 所示。

表 7-1 MATLAB 中的字符串查找函数

| 函 数 | 语 法 | 说 明 |
| --- | --- | --- |
| strfind | strfind(str,pattern) | 查找 str 中是否具有 pattern 子串，返回子串出现位置，没有出现则返回空数组 |
| findstr | findstr(str1,str2) | 查找 str1 和 str2 中，较短字符串在较长字符串中出现的位置，没有出现则返回空数组 |
| strmatch | strmatch(pattern,str)<br>strmatch(pattern,str,'exact') | 检查 pattern 是否和 str 最左侧的部分一致<br>检查 pattern 是否和 str 完全一致，相当于 strcmp |
| strtok | strtok(str,char) | 返回 str 中由 char 指定的字符前的部分和之后的部分，默认的 char 为空格、制表符或换行符 |

从表 7-1 中可以看出：

（1）strfind 函数中两个输入参数有先后之分，用于明确知道在哪个字符串中查找子串的情形；

（2）findstr 的两个输入参数没有先后分别，在对两个字符串长度未知的情况下使用；

（3）strmatch 实际上是比较 pattern 和 str 中前 length(pattern)个字符是否一致；

（4）strtok 可以用于从句子中分割截取单词。

**例 7-7** 字符串的查找。

**解：**在命令窗口输入：

```
>> a='Welcome to MATLAB!';
>> b='Welcome,Sir!';
>> c='Welcome';
>> ab=strcat(a,b)
ab =
Welcome to MATLAB!Welcome,Sir!
>> strfind(ab,c)      %strfind 中长字符串在前，否则返回空数组
ans =
     1    19
>> strfind(c,ab)
ans =
     []
>> findstr(ab,c)      %findstr 中两个字符串位置可变
ans =
     1    19
>> findstr(c,ab)
ans =
     1    19
>> strmatch(a,c)      %strmatch 中长字符串应在后
ans =
     []
>> strmatch(c,a)
ans =
     1
>> strmatch(c,a,'exact')
ans =
     []
>> [wel1,rem1]=strtok(ab,'!')
wel1 =
```



```
Welcome to MATLAB
rem1 =
!Welcome,Sir!
>> [wel2,rem2]=strtok(rem1,'!')
wel2 =
Welcome,Sir
rem2 =
!
```

## 7.2.3 其他操作

对于字符串数组，除了最常用的创建、比较和查找替换操作外，还有许多别的一些操作，MATLAB 提供了丰富的函数，如表 7-2、表 7-3 和表 7-4 所示。

表 7-2 空格处理函数

| 函 数 | 说 明 |
| --- | --- |
| blanks(n) | 创建由 n 个空格组成的字符串 |
| deblank(str) | 裁切字符串 str 的尾部空格 |
| strtrim(str) | 裁切字符串 str 的开头和尾部的空格、制表符和回车符 |

例 7-8 空格处理函数。

**解**：在命令窗口输入：

```
>> a=blanks(4)
a =

>> length(a)
ans =
     4
>> b='test    ';
>> length(b)
ans =
     8
>> c=deblank(b)
c =
test
>> length(c)
ans =
     4
>> d='   test    ';
>> length(d)
ans =
    11
>> e=strtrim(d)
e =
test
>> length(e)
ans =
     4
```

7.1 节中已经讲到，创建二维字符数组时，经常要用空格填满比较短的字符串，因此当提取每行字符串元素后，就需要删除字符串最右边的空格，deblank 函数常用来实现这一功能。

表 7-3　格式相关函数

| 函　数 | 说　明 |
|---|---|
| lower(str) | 将 str 中的字母全部转换成小写 |
| upper(str) | 将 str 中的字母全部转换成大写 |
| strjust(strarray,'right') | 将字符数组 strarray 的每一行右对齐 |
| strjust(strarray,'left') | 将字符数组 strarray 的每一行左对齐 |
| strjust(strarray,'center') | 将字符数组 strarray 的每一行居中对齐，其他位置用空格补齐 |
| sort(str) | 按照字符的 ASCII 值对字符串 str 排序 |

**例 7-9**　字符数组的格式操作。

**解**：在命令窗口输入：

```
>> a=strvcat('Welcome','to','MathWorks')
a =
Welcome
to
MathWorks
>> upper(a)
ans =
WELCOME
TO
MATHWORKS
>> strjust(a,'center')
ans =
 Welcome
   to
MathWorks
>> sort(a(3,:))        %对字符数组 a 的第三行字符串，按照升序排列
ans =
MWahkorst
>> sort(a)        %对字符数组 a 的每一列，按照字符升序排列
ans =
Ma
WelcWme
tothoorks
>> sort(a,2)       %对字符数组 a 的每一行，按照字符升序排列
ans =
  Wceelmo
       ot
MWahkorst
>> sort(a,2,'descend')  %对字符数组 a 的每一行，按照字符降序排列
ans =
omleecW
to
tsrokhaWM
```

需要注意的是，strjust 只能对字符数组进行对齐操作，不同于其他几个函数，可以对字符串组成的元胞数组（见 7.3）进行操作。

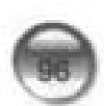




表 7-4 字符串的输入和输出

| 函 数 | 说 明 |
| --- | --- |
| sprintf | 按照指定格式将数据以字符串形式输出 |
| sscanf | 从标准输入接受字符串 |

表 7-4 中的两个函数格式控制都比较复杂，在程序设计中用于接收输入或输出到屏幕，从而达到与用户之间进行文字交互的功能，有兴趣的读者可以参考 MATLAB 的帮助。

## 7.3 字符串的元胞数组

把多个字符串存储起来，可以采用 7.1 介绍的创建字符串数组的方法，这样需要在较短的字符串末尾补空格。当然，还有一种方法就是采用元胞数组结构，在元胞数组的每一个元胞单元中，可以存储任意长度的字符串，这样，就不需要再在字符串末尾补空格了。创建以字符串为数据内容的元胞数组，和创建一般的元胞数组一样，请参考第 6 章的内容。

这里重复强调一下，元胞数组的每一个单元是一个元胞，元胞是 MATLAB 中一种特殊的结构，其中可以存储任何类型的数据。而以字符串为数据元素的元胞数组，就是在每一个元胞中存储了不同长度的字符串。

MATLAB 提供了字符数组和元胞数组之间的转换函数：

（1）cellstr 可以把字符数组每一行转换成元胞数组的一个元胞，转换时每行末尾的空格会被删除；

（2）char 函数能够把字符串元胞数组的每一个元胞的内容转换成字符串，然后纵向连接起来组成字符数组，转换时在较短的字符串末尾用空格补齐。

**例 7-10** 字符数组和字符串的元胞数组之间的转换。

**解**：在命令窗口输入：

```
>> data = ['Allison Jones';'Development  ';'Phoenix      '];
>> celldata = cellstr(data)
celldata =
    'Allison Jones'
    'Development'
    'Phoenix'
>> length(celldata{3})
ans =
     7
>> iscellstr(celldata) %测试变量是否属于字符串的元胞数组
ans =
     1
>> strings = char(celldata)
strings =
Allison Jones
Development
Phoenix
>> length(strings(3,:))
```

```
ans =
    13
```

对字符数组进行操作的函数大部分也可以用于字符串的元胞数组，表 7-5 列出对字符串元胞数组操作的函数的语法和说明。

表 7-5　字符串元胞数组操作函数

| 函数类 | 函　数 | 说　明 |
|---|---|---|
| 与字符数组的转换 | cellstr(strArray) | 把字符数组 strArray 每一行转换成一个元胞 |
| | char(strCell) | 把元胞数组 strCell 每一元胞转换成一行字符串，垂直连接成字符数组 |
| 连接 | strcat(*S1*,*S2*) | 把 *S1* 和 *S2* 对应元胞的字符串连接起来，不消除末尾空格 |
| 比较 | strcmp(*S*,*T*) | 比较 *S* 和 *T* 对应的元胞中字符串是否相同 |
| | strncmp(*S*,*T*,*n*) | 比较 *S* 和 *T* 对应的元胞中字符串前 *n* 个字符是否相同 |
| | strcmpi(*S*,*T*) | 忽略字母大小写，比较 *S* 和 *T* 对应的元胞中字符串是否相同 |
| | strncmpi(*S*,*T*) | 忽略字母大小写，比较 *S* 和 *T* 对应的元胞中字符串前 *n* 个字符是否相同 |
| 替换查找 | strrep(*S1*,*S2*,*S3*) | 对三个输入元胞数组的对应元胞做字符串替换 |
| | strfind(*S*,pattern) | 查找 *S* 的每一个元胞存储的字符串中 pattern 出现的位置 |
| | strmatch('str',*S*) | 查找 *S* 的每一个元胞存储的字符串是否以 str 开头（比字符数组的查找慢） |
| | strtok(*S*,'char') | 对 *S* 的每个元胞的字符串进行 strtok 操作，返回和 *S* 同尺寸的元胞数组 |
| 空格操作 | deblank(*S*) | 对 *S* 的每个元胞的字符串删除末尾空格 |
| | strtrim(*S*) | 对 *S* 的每个元胞的字符串删除开头和末尾空格、制表符和回车符 |
| 格式操作 | lower(*S*) | 把 *S* 的每个元胞的字符串中字符转换成小写 |
| | upper(*S*) | 把 *S* 的每个元胞的字符串中字符转换成大写 |
| | strjust(*S*,'style') | 清除 *S* 的每个元胞的字符串的空格后，再用空格补齐进行对齐操作 |
| | sort(*S*) | 对 *S* 的每个元胞的字符串进行指定的排序操作 |
| 测试 | iscellstr(*S*) | 测试 *S* 是否是字符串为内容的元胞数组 |

由表 7-5 可见，对字符数组的操作函数大部分也能用于操作字符串元胞数组，这时候是对元胞数组的每一个单元或对应单元内的字符串进行操作。不过，也有一些只能用于字符数组的函数，它们是：**strvcat**、**findstr**、**isletter**、**isspace**、**isstrprop** 和 **strjust**，容易看出这些函数都是需要对字符串中的字符进行单独处理的，因此，就很容易理解它们为何不能用于以元胞为单元的元胞数组。。

**例 7-11**　字符串元胞数组的操作。

**解**：在命令窗口输入：

```
>> a={'  a','bbbb';'c  ',' dd '}
a =
    '  a'    'bbbb'
    'c  '    ' dd '
>> b={'tt  ',' tt';' bb','bb  '}
b =
    'tt  '    ' tt'
    ' bb'    'bb  '
>> strcat(a,b)
ans =
```



```
    '   att   '    'bbbb  tt'
    'c     bb'    ' dd bb  '
>> strncmp(a,b,2)
ans =
     0     0
     0     0
>> c=strcat(a,b)
c =
    '   att   '    'bbbb  tt'
    'c     bb'    ' dd bb  '
>> strrep(c,'tt','TT')
ans =
    '   aTT   '    'bbbb  TT'
    'c     bb'    ' dd bb  '
>> deblank(c)
ans =
    '   att'      'bbbb  tt'
    'c     bb'    ' dd bb'
>> strtrim(c)
ans =
    'att'         'bbbb  tt'
    'c     bb'    'dd bb'
>> upper(c)
ans =
    '   ATT   '    'BBBB  TT'
    'C     BB'    ' DD BB  '
>> strjust(c,'left')
ans =
    'att      '    'bbbb  tt'
    'c     bb'    'dd bb   '
>> iscellstr(c)
ans =
     1
```

## 7.4 使用正则表达式搜索

除了前面提到的简单的查找替换函数，MATLAB 还支持用功能强大的正则表达式搜索。简单来说，正则表达式就是定义一种特定模式的字符串，在正则表达式中，经常用到通配符、标记符，用来指代特定的字符集或字符模式。例如，字符串'Joh?n\w*'就是一个 MATLAB 正则表达式，它表示一个具有下述模式特征的字符串：以 Jo 开头（Jo），接着可能是字符 h（h?），然后是字符 n（n），字符串最后以任意个非空格、制表符或回车符的字符（\w*）结尾。因此，Jon，John，Jonathan，Johnny 这些字符串就都能匹配这一正则表达式。

MATLAB 提供了三个函数用于支持正则表达式的搜索，如表 7-6 所示。正则表达式中可能应用到的通配符、标记符很多，函数返回结果也很复杂，有兴趣的读者请参考 MATLAB 帮助。

表 7-6 支持正则表达式的查找替换函数

| 函 数 | 功 能 |
|---|---|
| regexp('str', 'expr') | 搜索字符串 str 是否匹配正则表达式 expr |
| regexpi('str', 'expr') | 同 regexp，但忽略字母大小写 |
| regexprep('str', 'expr', 'repstr') | 用 repstr 替换字符串 str 中匹配正则表达式 expr 的部分 |

**例 7-12** 正则表达式的简单应用。

**解：** 在命令窗口输入：

```
>> stra = 'bat cat can car COAT court cut ct CAT-scan';
>> regexpi(stra, 'c[aeiou]+t') %匹配c*t模式的字符串，其中*是aeiou中的某一个字
符
ans =
     5    17    28    35
>> strb = {'Madrid, Spain' 'Romeo and Juliet' 'MATLAB is great'};
>> s1 = regexp(strb, '[A-Z]');  %匹配大写字母
>> s2 = regexp(strb, '\s');     %匹配空格字符
>> celldisp(s1)
s1{1} =
     1     9
s1{2} =
     1    11
s1{3} =
     1     2     3     4     5     6
>> celldisp(s2)
s2{1} =
     8
s2{2} =
     6    10
s2{3} =
     7    10
```

正则表达式功能强大，经常用在大文本的操作中，这里只是简单地提及 MATLAB 中的几个函数，并给出了一个简单的例子。MATLAB 的联机帮助中，有对正则表达式中各种标记字符和上述三个函数输出结果的详细介绍，请读者自己参阅。

## 7.5 字符数组和数值数组间的相互转换

本章开头已经讲到，MATLAB 中字符类型的内部运算都是按照其 Unicode 编码对应的数字进行的，因此字符数组和数值数组之间的转换就很容易了。而且，在图形绘制的标注中，经常要用到 MATLAB 中提供的转换函数如表 7-7 和表 7-8 所示。

表 7-7 数值数组转换成字符数组的函数

| 函 数 | 说 明 |
|---|---|
| char | 把正整数转换成对应的 Unicode 字符（忽略小数部分） |
| int2str | 把数值数组转换成整数数字组成的字符数组（小数部分 round 取整） |



续表

| 函　　数 | 说　　明 |
|---|---|
| num2str | 按指定精度和格式把数字转换成字符串 |
| mat2str | 将数组转换成字符串，可以指定精度和是否带类型名 |
| dec2bin | 将十进制正整数转换成二进制，以字符串输出 |
| dec2hex | 将十进制正整数转换成十六进制，以字符串输出 |
| dec2base | 将十进制正整数转换成指定进制，以字符串输出 |

表 7-8　字符数组转换成数值数组的函数

| 函　　数 | 说　　明 |
|---|---|
| unitN | 将字符串转换成 Unicode 编码对应的数值 |
| str2num | 将数字字符串转换成代表的数字 |
| str2double | 将数字字符串或数字字符串的元胞数组，转换成对应的双精度浮点数 |
| bin2dec | 将二进制的字符串转换成对应的十进制数字 |
| hex2dec | 将十六进制的字符串转换成对应的十进制数字 |
| base2dec | 将指定进制的字符串转换成对应的十进制数字 |

**例 7-13**　把数值数组转换成字符数组。

解：在命令窗口输入：

```
>> x = [77 65 84 76 65 66];
>> xx=char(x)
xx =
MATLAB
>> y=int2str(rand(3))
y =
0  1  0
1  1  1
0  1  1
>> z=mat2str([1 2 3.5])
z =
[1 2 3.5]
>> a=dec2bin(356)
a =
101100100
>> b=dec2base(356,7)
b =
1016
>> whos
  Name      Size                    Bytes  Class

  a         1x9                        18  char array
  b         1x4                         8  char array
  x         1x6                        48  double array
  xx        1x6                        12  char array
  y         3x7                        42  char array
  z         1x9                        18  char array

Grand total is 55 elements using 146 bytes
```

例 7-14 把字符数组转换成数值数组。

解：在命令窗口输入：

```
>> name = 'Thomas R. Lee';
>> name = double(name)
name =
    84   104   111   109    97   115    32    82    46    32    76   101   101
>> name = char(name)
name =
Thomas R. Lee
>> str = '37.294e-1';
>> val = str2num(str)
val =
    3.7294
>> c = {'37.294e-1'; '-58.375'; '13.796'};
>> d = str2double(c)
d =
    3.7294
  -58.3750
   13.7960
>> whos c d
  Name      Size                    Bytes  Class

  c         3x1                       224  cell array
  d         3x1                        24  double array

Grand total is 28 elements using 248 bytes

>> hex2dec('3ff')
ans =
        1023
>> bin2dec('010111')
ans =
    23
>> base2dec('212',3)
ans =
    23
```

## 7.6 小结

本章介绍了 MATLAB 中又一种数据类型——字符串，重点讲解了字符数组和字符串元胞数组的创建、比较、替换查找等多种操作方法，以及字符数组和数值数组之间的转换。这些是本章的重点内容，读者应熟练掌握。此外，本章还简单介绍了 MATLAB 中功能强大的正则表达式，供有兴趣的读者参考。

# 第 8 章 关系运算和逻辑运算

MATLAB 中的运算包括算术运算、关系运算和逻辑运算。关系运算和逻辑运算在程序设计中应用十分广泛。算术运算用于数值计算，关系运算则是用于比较两个操作数，而逻辑运算则是对简单逻辑表达式进行复合的运算。关系运算和逻辑运算的返回结果都是逻辑类型（1 代表逻辑真，0 代表逻辑假）。

本章主要讲解 MATLAB 中的关系运算和逻辑运算，特别强调了其中容易出错的地方，最后还简单介绍了逻辑函数、测试函数以及运算优先级等知识。

## 8.1 逻辑类型的数据

逻辑类型的数据经常用在程序流程控制中，尤其是在分支条件的选择和循环终止条件的确定上。逻辑类型数据只有逻辑真和逻辑假两个，MATLAB 中用 1 代表逻辑真，用 0 代表逻辑假，用 true 和 false 函数可以创建逻辑类型的数据。

**例 8-1** 逻辑类型数据。

**解**：在命令窗口输入：

```
>> x=1
x =
     1
>> xx=true  %创建逻辑类型的变量 xx，并对其赋值为逻辑真
xx =
     1
>> yy=false
yy =
     0
>> whos
```

```
  Name      Size                    Bytes  Class

  x         1x1                         8  double array
  xx        1x1                         1  logical array
  yy        1x1                         1  logical array

Grand total is 3 elements using 10 bytes
```

需要注意的是，逻辑类型的 1/0 和浮点类型的 1/0 是不同的。例 8-1 中通过 whos 显示工作区中变量的详细信息，读者就可以看到，true 创建了一个逻辑型的数据，内存中占用 1 个字节（8 位），而数值类型的 1（默认是双精度浮点类型）则需要占用 8 个字节（64 位）。不过，实际使用中这一点区别影响不大，在逻辑运算中，MATLAB 把所有非 0 的数值都当做逻辑真（true）处理。

## 8.2 关系运算

关系运算用于比较两个操作数的大小关系，并返回逻辑真（1）或者逻辑假（0），MATLAB 中的关系运算符如表 8-1 所示。

表 8-1 关系运算符

| 关系运算符 | 说 明 | 举 例 |
|---|---|---|
| < | 小于 | 3>2 返回逻辑真（1） |
| <= | 小于等于 | 3<=2 返回逻辑假（0） |
| > | 大于 | 2<2 返回逻辑假（0） |
| >= | 大于等于 | 2>=2 返回逻辑真（1） |
| == | 等于 | 2==2 返回逻辑真（1） |
| ~= | 不等于 | 2~=2 返回逻辑假（0） |

当操作数是数组形式时，关系运算符总是对被比较的两个数组的各个对应元素进行比较，因此要求被比较的数组必须具有相同的尺寸。

**例 8-2** MATLAB 中的关系运算。

**解：** 在命令窗口输入：

```
>> 3>=2
ans =
     1
>> a=rand(2,3)
a =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
>> b=rand(2,3)
b =
    0.4565    0.8214    0.6154
    0.0185    0.4447    0.7919
>> a>b
ans =
```



```
     1     0     1
     1     1     0
```

需要注意的是：

（1）比较两个数是否相等的关系运算符是两个等号"=="，而单个的等号"="在 MATLAB 中是变量赋值符号；

（2）比较两个浮点数是否相等时需要注意，由于浮点数的存储形式决定的相对误差的存在，在程序设计中最好不要直接比较两个浮点数是否相等，而是采用大于、小于的比较运算将待确定值限制在一个满足需要的区间之内。

**例 8-3**　浮点数的比较运算。

**解**：在命令窗口输入：

```
>> 3==3+eps %比较发现，3 和 3+eps 相等
ans =
     1
>> 3==3-eps
ans =
     1
>> 3<3+1*10^-6        %实际使用中，将待确定的浮点数限制在某个满足需要的区间即可
ans =
     1
>> 3>3-1*10^-6
ans =
     1
>> 3<3+eps
ans =
     0
>> 3>3-eps
ans =
     0
```

## 8.3　逻辑运算

关系运算返回的结果是逻辑类型（逻辑真或逻辑假），这些简单的逻辑数据可以通过逻辑运算符组合形成复杂的逻辑表达式，这在程序设计中经常用于进行分支选择或者确定循环终止条件。

MATLAB 中的逻辑运算有 3 类：

（1）逐个元素的逻辑运算；

（2）捷径逻辑运算；

（3）逐位逻辑运算。

只有前两种逻辑运算返回逻辑类型的结果。

### 8.3.1　逐个元素的逻辑运算

逐个元素的逻辑运算符有三种：逻辑与（&）、逻辑或（|）和逻辑非（~）。前两个是双

目运算符，必须有两个操作数参与运算，逻辑非是单目运算符，只对单个元素进行运算。其意义和示例如表 8-2 所示。

表 8-2 逐个元素的逻辑运算符

| 运算符 | 说明 | 举例 |
| --- | --- | --- |
| & | 逻辑与：双目逻辑运算符<br>参与运算的两个元素都为逻辑真或非零时，返回逻辑真（1），否则返回逻辑假（0） | 1&true 返回 1（true）<br>1&0 返回 0（false）<br>0&0 返回 0（false） |
| \| | 逻辑或：双目逻辑运算符<br>参与运算的两个元素都为逻辑假或零时，返回逻辑假（0），否则返回逻辑真（1） | 0\|0 返回 0（false）<br>1\|0 返回 1（true）<br>1\|1 返回 1（true） |
| ~ | 逻辑非：单目逻辑运算符<br>参与运算的元素为逻辑真或非零时，返回逻辑假（0），否则返回逻辑真（1） | ~1 返回 0（false）<br>~0 返回 1（true） |

这里的逻辑与和逻辑非运算，都是逐个元素进行双目运算，因此如果参与运算的是数组，就要求两个数组具有相同的尺寸。

**例 8-4** 逐个元素的逻辑运算。

解：在命令窗口输入：

```
>> A=rand(2,4)
A =
    0.9218    0.1763    0.9355    0.4103
    0.7382    0.4057    0.9169    0.8936
>> B=A>0.5  %数组 A 中大于 0.5 的元素位置上，将数组 B 赋值为逻辑真，否则赋值为逻辑假
B =
     1     0     1     0
     1     0     1     1
>> C=A<0.7
C =
     0     1     0     1
     0     1     0     0
>> B&C
ans =
     0     0     0     0
     0     0     0     0
>> B|C
ans =
     1     1     1     1
     1     1     1     1
>> ~B
ans =
     0     1     0     1
     0     1     0     0
>> A&C  %所有非零的数值类型都被当做逻辑真处理
ans =
     0     1     0     1
     0     1     0     0
>> A|C
```



```
ans =
     1     1     1     1
     1     1     1     1
>> ~A
ans =
     0     0     0     0
     0     0     0     0
```

### 8.3.2 捷径逻辑运算

MATLAB 中的捷径逻辑运算符有两个：逻辑与（&&）和逻辑或（||）。实际上它们的运算功能和前面讲过的逐个元素的逻辑运算符相似，只不过在一些特殊情况下，捷径逻辑运算符会减少一些逻辑判断的操作。

当参与逻辑与运算的两个数据都同为逻辑真（非零）时，逻辑与运算才返回逻辑真（1），否则都返回逻辑假（0）。

&&运算符就是利用这一特点，当参与运算的第一个操作数为逻辑假时，直接返回假，而不再去计算第二个操作数。

&运算符在任何情况下都要计算两个操作数的结果，然后取逻辑与。

||的情况类似，当第一个操作数为逻辑真时，||直接返回逻辑真，而不再去计算第二个操作数。

|运算符任何情况下都要计算两个操作数的结果，然后取逻辑或。

捷径逻辑运算符如表 8-3 所示。

表 8-3 捷径逻辑运算符

| 运 算 符 | 说 明 |
|---|---|
| && | 逻辑与：当第一个操作数为假，直接返回假，否则同& |
| \|\| | 逻辑或：当第一个操作数为真，直接返回真，否则同\| |

因此，捷径逻辑运算符比相应的逐个元素的逻辑运算符的运算效率更高，在实际编程中，一般都使用捷径逻辑运算符。而且当运算中第二个表达式依赖于第一个条件成立时，则只能使用捷径逻辑运算符，否则可能报错。如例 8-5 中用&的话，则当 $b$ 等于 0 时，会出现警告提示。

**例 8-5** 捷径逻辑运算符。

**解**：在命令窗口输入：

```
>> b=0
b =
     0
>> (b~=0)&&(18/b>3)
ans =
     0
>> (b~=0)&(18/b>3)
Warning: Divide by zero.
ans =
     0
```

### 8.3.3 逐位逻辑运算

逐位逻辑运算能够对非负整数的二进制形式进行逐位逻辑运算，并将逐位运算后的二进制数值转换成十进制数值输出。MATLAB 中的逐位逻辑运算函数如表 8-4 所示。

表 8-4 逐位逻辑运算函数

| 函 数 | 说 明 |
| --- | --- |
| bitand(*a*,*b*) | 逐位逻辑与，*a* 和 *b* 的二进制数位上都为 1 则返回 1，否则返回 0，并把逐位逻辑运算后的二进制数字转换成十进制数值输出 |
| bitor(*a*,*b*) | 逐位逻辑或，*a* 和 *b* 的二进制数位上都为 0 则返回 0，否则返回 1，并把逐位逻辑运算后的二进制数字转换成十进制数值输出 |
| bitcmp(*a*,*n*) | 逐位逻辑非，将数字 *a* 扩展成 *n* 位二进制形式，当扩展后的二进制数位上为 1 则返回 0，否则返回 1，并把逐位逻辑运算非后的二进制数字转换成十进制数值输出 |
| bitxor(*a*,*b*) | 逐位逻辑异或，*a* 和 *b* 对应的二进制数位上相同则返回 0，否则返回 1，并把逐位逻辑运算后的二进制数字转换成十进制数值输出 |

**例 8-6** 逐位逻辑运算函数。

**解：**在命令窗口输入：

```
>> a=25;b=23;
>> aa=bitxor(a,b);
>> dec2bin(a)          %将十进制数字 a 转换成二进制形式，并以字符串格式输出
ans =
11001
>> dec2bin(b)
ans =
10111
>> dec2bin(aa)
ans =
1110
>> dec2bin(bitand(a,b))
ans =
10001
>> dec2bin(bitor(a,b))
ans =
11111
>> dec2bin(bitcmp(a,6))
ans =
100110
>> whos
  Name      Size                    Bytes  Class

  a         1x1                         8  double array
  aa        1x1                         8  double array
  ans       1x6                        12  char array
  b         1x1                         8  double array

Grand total is 9 elements using 36 bytes
```





## 8.4 逻辑函数和测试函数

在逻辑运算方面，除了8.3节介绍的逻辑运算符和逐位逻辑运算函数外，MATLAB还提供了许多逻辑运算函数，如表8-5所示。

表8-5 逻辑运算函数

| 函　数 | 说　明 |
| --- | --- |
| xor(*A*,*B*) | 异或函数：当*A*和*B*对应位置上不同为真（非零）且不同为假（零）时，返回真（1），否则返回假（0） |
| all(*A*) | 当*A*为一维数组时，若其元素全部为逻辑真（非零），则返回真，否则返回假<br>当*A*为二维数组时，若其一列上的所有元素都非零，则对应列的位置返回真，否则返回假 |
| any(*A*) | 当*A*为一维数组时，若其元素不全部为逻辑假（零），则返回真，否则返回假<br>当*A*为二维数组时，若其一列上的所有元素不全为零，则对应列的位置返回真，否则返回假 |

**例8-7** MATLAB中的逻辑运算函数。

**解：** 在命令窗口输入：

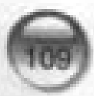
```
>> A=rand(2,4)
A =
    0.2722    0.0153    0.4451    0.4660
    0.1988    0.7468    0.9318    0.4186
>> B=A>0.5
B =
     0     0     0     0
     0     1     1     0
>> C=A<0.9
C =
     1     1     1     1
     1     1     0     1
>> xor(B,C)
ans =
     1     1     1     1
     1     0     1     1
>> all(C)
ans =
     1     1     0     1
>> any(B)
ans =
     0     1     1     0
```

在关系运算方面，只有8.2节介绍的关系运算符往往是不够用的，甚至可能引起混乱。比如在数组操作中经常遇到的空数组和非数值型（NaN）元素，使用8.2节介绍的常规关系运算符就可能出现错误。

**例8-8** 空数组和非数值型（NaN）元素参与的关系运算。

**解：** 在命令窗口输入：

```
>> a=[];
>> a==[]
ans =
```



```
    []
>> b=NaN;c=NaN;
>> b==c
ans =
     0
```

例 8-8 出现的结果，是因为等于的比较运算是逐个元素比较的，而空数组 $a$ 中没有元素，因此 $a$ 与空数组比较后返回结果也是一个没有元素的空数组。$b$ 和 $c$ 都是非数值型(NaN)的元素，前面章节已经讲过，任何两个 NaN 都是互不相同的，因此 $b$ 和 $c$ 等于的比较，返回结果为逻辑假。

为了处理这一类问题，MATLAB 提供了丰富的测试函数，这些测试函数都以 is 开头，常用的测试函数如表 8-6 所示。

表 8-6　测试函数

| 测试函数 | 功　能 |
|---|---|
| isa | 测试待测变量是否属于某个指定的类型 |
| iscell | 测试待测变量是否为元胞数组类型 |
| iscellstr | 测试待测变量是否为元胞数组或结构体类型 |
| ischar | 测试待测变量是否为字符串类型 |
| isdir | 测试待测变量是否为目录 |
| isempty | 测试待测变量是否为空数组 |
| isequal | 测试待测变量是否相等 |
| isequalwithequalnans | 测试待测变量是否相等（认为所有 NaN 相等） |
| isfield | 测试待测变量是否是指定结构体的某个字段 |
| isfinite | 测试待测变量是否有限 |
| isfloat | 测试待测变量是否为浮点类型 |
| isglobal | 测试待测变量是否为全局变量 |
| isinf | 测试待测变量是否无限 |
| isinteger | 测试待测变量是否为整数类型 |
| iskeyword | 测试待测变量是否为 MATLAB 关键字 |
| islogical | 测试待测变量是否为逻辑类型 |
| ismember | 测试待测变量是否为指定集合的成员 |
| isnan | 测试待测变量是否为非数值量（NaN） |
| isnumeric | 测试待测变量是否为数值类型 |
| isreal | 测试待测变量是否为实数 |
| isscalar | 测试待测变量是否为 1*1 数组 |
| issorted | 测试待测变量是否已经有序排列 |
| isspace | 测试待测变量是否为空格 |
| issparse | 测试待测变量是否为稀疏矩阵 |
| isstruct | 测试待测变量是否为结构体 |
| isvarname | 测试待测变量是否为 MATLAB 合法变量名 |
| isvector | 测试待测变量是否为一维数组 |



**例 8-9**　测试函数的应用。

**解：** 在命令窗口输入：

```
>> a=[];
>> isempty(a)
ans =
     1
>> b=NaN;
>> isnan(b)
ans =
     1
>> c=[1 0 NaN];d=c;
>> isequal(c,c)
ans =
     0
>> isequal(c,d)
ans =
     0
>> isequalwithequalnans(c,d)
ans =
     1
>> isequalwithequalnans(c,c)
ans =
     1
```

## 8.5　运算优先级

在 MATLAB 程序中，经常出现一个表达式中需要处理多种不同类型运算的情况，这时就需要考虑运算的优先级问题。

表 8-7 按照运算优先级从高到低的顺序列出了 MATLAB 中的各种运算。

**表 8-7　MATLAB 中的运算优先级**

| 优先级（数字越小，优先级越高） | 运算及其说明 |
|---|---|
| 1 | 括号（()） |
| 2 | 转置和乘幂（.'，.^，'，^） |
| 3 | 一元加/减运算（+，-），逻辑非（~） |
| 4 | 乘（*），除（/，\），点乘（.*），点除（./，.\） |
| 5 | 加（+），减（-） |
| 6 | 冒号运算（:） |
| 7 | 关系运算（>，>=，<，<=，==，~=） |
| 8 | 逐个元素的逻辑与（&） |
| 9 | 逐个元素的逻辑或（\|） |
| 10 | 捷径逻辑与（&&） |
| 11 | 捷径逻辑或（\|\|） |

由表 8-7 可见，括号运算优先级最高，其次是各类算术运算，然后是关系运算，优先级最低的是逻辑运算。

在实际的表达式书写中，建议尽量采用括号分隔的方式明确各步运算的次序，以尽可能地减少优先级混乱。比如 ***a***./***b***.^2 最好写成 ***a***./(***b***.^2)，等等。

## 8.6 小结

通过本章的学习，读者应掌握 MATLAB 中的关系运算和逻辑运算，尤其是其中容易引起混乱和错误的地方更要熟悉掌握。此外，读者还应该对 MATLAB 中的逻辑型数据类型有比较深入的认识，为将来编写 MATLAB 代码打下好的基础。本章的结尾部分还介绍了 MATLAB 中各类运算的优先级问题，一个好的习惯是通过括号分隔的方法避免表达式中出现可能有歧义的运算次序。



# 第 9 章

# 程序控制流

和各种常见的高级语言一样，MATLAB 也提供了多种经典的程序结构控制语句。本章介绍 MATLAB 中的四大类程序流程控制语句：分支控制语句（if 结构和 switch 结构）、循环控制语句（for 循环、while 循环、continue 语句和 break 语句）、错误控制语句（try-catch 结构）和程序终止语句（return 语句）。

## 9.1 分支控制语句

分支语句可以使程序中的一段代码只在满足一定条件时才执行，因此也称为分支选择。MATLAB 中分支语句有两类：if 语句和 switch 语句。

（1）if 与 else 或 elseif 连用，偏向于是非选择，当某个逻辑条件满足时，执行 if 后的语句，否则执行 else 语句；

（2）switch 和 case、otherwise 连用，偏向于情况的列举，当表达式结果为某个或某些值时，执行特定 case 指定的语句段，否则执行 otherwise 语句。

### 9.1.1 if，else 和 elseif

if 结构的语法形式如下：

```
if logical_expression
   statements
elseif logical_expression
         statements
else logical_expression
         statements
end
```

其中 elseif 和 else 语句都是可选语句。if，elseif 和 else 构成的各项分支里面，哪个的条件满足（逻辑表达式 logical_expression 的结果为真），就执行哪个分支后面紧跟的程序语句。因此，各条分支条件满足的情况应该是互斥的和完备的，就是被选的条件能在一个分支中成立，而且只能在一个分支中成立。

当然，省略了 elseif 和 else 分支的语句，就不必要求分支条件满足的情况具备完备性了。

**例 9-1** if 结构。

**解：**在命令窗口输入：

```
%Ex0901      if 结构
clear    %清除工作区的所有变量
a=7;
if rem(a,2)==0  %当 a 能够被 2 整除时，显示'a 是偶数'
   disp(strcat(num2str(a),'是偶数'));
else     %否则，即 a 不能被 2 整除时，显示'a 是偶数'
   disp(strcat(num2str(a),'是奇数'));
end
```

运行结果是：

```
7 是奇数。
```

if 结构中条件判断除了可以用逻辑表达式外，还可以用数组 $A$，这时判断相当于逻辑表达式 all($A$)，即当数组 $A$ 的所有元素都非零，才执行该条件后的分支代码。

特别的，当数组 $A$ 为空数组时，相当于该条件判断为假。

**例 9-2** 数组用于 if 结构。

**解：**在命令窗口输入：

```
%Ex0902      数组用于 if 结构
clear
A=zeros(3);
if A
   disp('全零数组被判为逻辑真');
else
   disp('全零数组被判为逻辑假');
end
B=[];
if B
   disp('空数组被判为逻辑真');
else
   disp('空数组被判为逻辑假');
end
```

运行结果是：

```
全零数组被判为逻辑假
空数组被判为逻辑假
```





### 9.1.2　switch，case 和 otherwise

switch 结构的语法形式是：

```
switch expression (scalar or string)
   case value1
      statements      % Executes if expression is value1
   case value2
      statements      % Executes if expression is value2
   .
   .
   .
   otherwise
      statements      % Executes if expression does not
                      % match any case
end
```

执行中，先计算表达式 expression 的值，当结果等于某个 case 的 value 时，就执行该 case 紧随的语句。如果所有 case 的 value 都和 expression 计算结果不相等，则执行 otherwise 后面紧随的语句。otherwise 语句是可选的，当没有 otherwise 语句时，如果所有 case 的 value 都和 expression 计算结果不相等，则直接执行后续代码。

switch 语句与 if 语句具有相等价的意义，对于数值类型来说，相当于判断 if result==value，对于字符串类型来说，相当于判断 if strcmp(result,value)。

由此可见，switch-case 语句实际上可以被 if-elseif-else 语句等效替换，不过两种结构各有便利的地方，读者在以后的例子中会逐渐体会到。

**例 9-3**　switch-case 结构。

**解**：在命令窗口输入：

```
%Ex0903 switch-case 结构
clear
a=4;
switch rem(a,2)
    case 0
        disp([num2str(a),'是偶数']);
    case 1
        disp([num2str(a),'是奇数']);
end
```

运行结果是：

```
4 是偶数。
```

学过 C 语言的读者需要注意，MATLAB 中的 switch-case 结构，只执行表达式结果匹配的第一个 case 分支，然后就跳出 switch-case 结构。因此，在每一个 case 语句中不需要用 break 语句跳出。

在一条 case 语句后可以列举多个值，只需要以元胞数组的形式列举多个值，就是用花括号把用逗号或空格分隔的多个值括起来即可。

**例 9-4**　一条 case 语句列举多个值的 switch-case 语句。



解：在命令窗口输入：

```
%Ex0904 一条case语句列举多个值的switch-case语句
clear
var=4;
switch var
    case 1
        disp('var=1');
    case {2,3,4}
        disp('var=2 or 3 or 4');
    otherwise
        disp('no match detected');
end
```

运行结果是：

```
var=2 or 3 or 4.
```

## 9.2 循环控制语句

循环控制语句能够使得某段程序代码多次重复执行，MATLAB 中提供了两类循环语句，分别是 for 循环和 while 循环：

（1）for 循环一般用在已知循环执行次数的情况；

（2）while 循环则用在已知循环退出条件的情况。

MATLAB 还提供了 continue 和 break 语句，用于更精细地控制循环结构：

（1）continue 语句使得当前次循环不向下执行，直接进入下一次循环；

（2）break 语句则是完全退出该循环。

### 9.2.1 for 循环

for 循环用于知道循环次数的情况，其语法格式为：

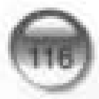
```
for index = start:increment:end
    statements
end
```

index 为循环变量，increment 为增量，end 用于判断循环是否应该终止。增量 increment 默认为 1，可以自由设置，当增量为正数时，循环开始先将 index 赋值为 start，然后判断 index 是否小于等于 end，若是，则执行循环语句，执行完后，对 index 累加一个增量 increment，再判断 index 是否小于等于 end，若是，则继续执行循环语句，对 index 累加，循环往复，直到 index 大于 end 时退出循环。

增量 increment 也可以设置为负整数，表示每次循环执行后对循环变量 index 进行递减，当 index 小于 end 时，退出循环。

**例 9-5** for 循环。

解：在命令窗口输入：





```
%Ex0905      for 循环,计算 1-100 所有整数的和
clear
result=0;
for i=1:100
    result=result+i;
end
result
```

运行结果为：

```
result =
        5050
```

MATLAB 中，循环的执行效率很低，提高程序效率的一个办法就是多采用数组结构和 MATLAB 内联函数。如例 9-5 则可以写成 result=sum（1：100），则效率更高。

for 循环中的循环变量 index 也可以赋值为数组 ***A***，那么在第一次循环中 index 就被赋值为 ***A***（：,1），即 ***A*** 的第一列元素，第二次循环 index 被赋值为 ***A***（：,2），以此类推，若 ***A*** 有 *n* 列元素，则循环执行 *n* 次，第 *n* 次循环时，循环变量 index 被赋值为 ***A***（：,*n*）。

**例 9-6** 数组赋值循环变量的 for 循环。

**解**：在命令窗口输入：

```
%Ex0906      数组赋值循环变量的 for 循环
clear
A=rand(2,4)
i=0;
for k=A
    i=i+1;
    disp(['-----loop',num2str(i),'-----']);
    k
end
```

运行结果为：

```
A =
    0.9501    0.6068    0.8913    0.4565
    0.2311    0.4860    0.7621    0.0185
-----loop1-----
k =
    0.9501
    0.2311
-----loop2-----
k =
    0.6068
    0.4860
-----loop3-----
k =
    0.8913
    0.7621
-----loop4-----
k =
    0.4565
    0.0185
```

### 9.2.2 while 循环

while 循环用于已知循环退出条件的情况，其语法形式如下：

```
while expression
   statements
end
```

当表达式 expression 的结果为真时，就执行循环语句，直到表达式 expression 的结果为假，才退出循环。

如果表达式 expression 是一个数组 ***A***，则相当于判断 all(***A***)。特别的，空数组则被当做逻辑假，循环不执行。

**例 9-7** while 循环。

**解：**在命令窗口输入：

```
%Ex0907    while 循环，计算前 n 个自然数的阶乘，直到超过 30 位
clear
k=0;
result=1;
while result<=1e30
   k=k+1;
   result=result*k;
end
disp(['1～',num2str(k),'的阶乘刚刚超过 30 位，是',num2str(length(num2str
(result))),'位'])
disp(['result=',num2str(result)])
```

运行结果是：

```
1～29 的阶乘刚刚超过 30 位，是 31 位
result=8841761993739700000000000000000
```

### 9.2.3 continue 语句

continue 语句用在循环中，表示当前次循环不再继续向下执行，而是直接对循环变量进行递增，进入下一次循环。

**例 9-8** continue 语句。

**解：**在命令窗口输入：

```
%Ex0908    continue 语句    提取字符串中所有字母
clear
str='MATLAB R14.3 version';
result=[];
for i=1:length(str)
   if(~isletter(str))
      continue;
   end
   result=[result,str(i)];
end
result
```



例 9-8 中用 for 循环遍历字符串 str 的所有字符，若当前字符不是字母，continue 语句直接跳出当次循环，否则向 result 结果中写入字符串中当前位置的字母。运行结果是：

```
result =
MATLAB R14.3 version
```

前面已经讲到，MATLAB 执行循环效率很低，尤其遇到多重循环和循环次数很大的时候，例 9-8 实际上可以通过非循环的数组函数直接实现：

str(find(isletter(str)==1))，isletter 测试字符数组中各个字符是否为字母，并返回和 str 同尺寸的逻辑数组。

find 查找这个逻辑数组中取值为真的元素的索引，然后通过索引访问，裁取 str 字符串中所有字母。这样运行效率会高一些。

## 9.2.4 break 语句

break 语句用于退出循环。

**例 9-9** break 语句。

**解**：在命令窗口输入：

```
%Ex0909    break 语句      查找二维数组每行中第一个零元素所在列位置
clear;
clc     %清除命令窗口显示
m=3;n=4
A=rand(m,n)<0.7
result=zeros(m,1);
for i=1:m
    for j=1:n
        if ~A(i,j)
            result(i)=j;
            break;
        end
    end
    if result(i)==0
        result(i)=Inf;
    end
end
result
```

例 9-9 实现了查找某二维逻辑数组 ***A***（元素为 0 或 1）中每一行上第一个零元素的位置。rand(m,*n*)<0.7 语句通过关系运算返回一个二维逻辑数组，两层 for 循环遍历这一数组的每一行每一列，在某一行中找到非零元素时，将其下标保存在结果中，并退出这一行在列方向的循环（注：break 仅仅是退出它所在的那一层循环）。

该程序某次的运行结果为：

```
A =
     1     1     1     0
     1     1     1     0
     1     1     1     1
```

```
result =
     4
     4
   Inf
```

### 9.2.5 数组结构和循环的效率比较

MATLAB 中执行循环的效率很低，在实际编程中，能通过数组结构和内部函数实现的操作，最好不要用循环来实现，这样能提高程序代码的执行效率。例 9-10 通过程序计时函数 tic/toc 比较了 MATLAB 中用循环和数组函数实现相同功能时性能上的差异。

**例 9-10** 循环和数组函数效率比较。

**解：** 在命令窗口输入：

```
%Ex0910    MATLAB 中的循环执行效率
clear
clc
t=0:0.01:2*pi;
disp('-----loop result-----');
tic
for i=1:length(t)
   y(i)=sin(t(i));
end
toc
disp('-----vectorizing-----');
tic
y=sin(t);
```

toc 其运行结果是：

```
-----loop result-----
Elapsed time is 0.001517 seconds.
-----vectorizing-----
Elapsed time is 0.000201 seconds.
```

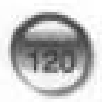
从运行结果可见，利用循环的效率明显是不如用 MATLAB 的内部数组函数的。

## 9.3 错误控制的 try-catch 结构

在程序设计中，有时候会遇到不能确定某段代码是否会出现运行错误的情况。这时候就可以用错误控制结构了。MATLAB 提供了 try-catch 结构用来捕获和处理错误。

try-catch 结构的语法格式是：

```
try
   statement
   …
   statement
catch
   statement
   …
   statement
end
```





程序运行时，首先尝试执行 try 语句后面的代码段，如果 try 和 catch 之间的代码执行没有错误发生，则程序通过，不执行 catch 和 end 之间的部分，而是继续执行 end 后面的代码。一旦 try 和 catch 之间的代码执行发生错误，则立刻转而执行 catch 和 end 之间的部分，然后才继续执行 end 后的代码。MATLAB 提供了 lasterr 函数，可以获取出错的原因。

**例 9-11**　try-catch 结构。

**解：** 在命令窗口输入：

```
%Ex0911     try-catch 结构
clear
clc
try
   det(rand(2,4));         %人造错误，求行列式要求数组行列数相等
   disp('no error found');
catch
   disp('catch codon is executed');
   disp('错误是：')
   disp(lasterr)
end
```

运行结果为：

```
catch codon is executed
错误是：
Error using ==> det
Matrix must be square.
```

如果删掉人为错误行，运行结果则为：no error found。

## 9.4　程序终止的 return 语句

程序代码一般按流程执行完毕后正常退出，但当遇到某些特殊情况，程序需要立即退出时，就可以用 return 提前终止程序运行。

**例 9-12**　return 语句。

**解：** 在命令窗口输入：

```
%Ex0912     return 语句
clear
clc
n=-2;
if n<0
   disp('negative number!');
   return;
end
disp('codon after return')
```

例 9-12 中当变量 $n$ 取值为负数时，通过 return 直接退出，不执行 if 后的代码。其运行结果是：

```
negative number!
```

若去掉其中的 return 语句，则运行结果变为：

```
negative number!
codon after return
```

return 语句更多的用在 MATLAB 函数 M 文件中，在后续的章节中会有实际的举例。

## 9.5 小结

本章介绍了 MATLAB 中的四大类程序流程控制语句，其中分支选择结构和循环结构是本章的重点。这些基础的程序流程结构，经常地被用在 MATLAB 的脚本 M 文件和函数 M 文件中，通过这些结构元件可以设计出复杂功能的 MATLAB 代码，后续章节中会经常碰到实际例子，读者一定要熟悉每一种程序流程的执行原理和本章提到的注意事项。

另外，关于错误捕获和处理，MATLAB 中还有许多内部函数和方法，本章只是简要地介绍了 try-catch-end 结构，有兴趣的读者可以参阅 MATLAB 联机帮助获得更详细的说明。

# 第 10 章 函数

本书从第 3 章到第 9 章，已经详细地介绍了 MATLAB 中的各种基本数据类型和程序流控制语句，本章在此基础上讲述 MATLAB 编程的知识。

## 10.1 M 文件和 MATLAB 编程概述

### 10.1.1 M 文件概述

MATLAB 提供了极其丰富的内部函数，使得用户通过命令行调用就可完成很多工作，但要想更加高效地利用 MATLAB，离不开 MATLAB 编程。用户可以通过组织一个 MATLAB 命令序列完成一个独立的功能，这就是脚本 M 文件编程；而把 M 文件抽象封装，形成可以重复利用的功能块，这就是函数 M 文件编程。因此，MATLAB 编程是提高 MATLAB 应用效率，把 MATLAB 基本函数扩展为实际的用户应用的必经之道。

M 文件是包含 MATLAB 代码的文件。M 文件按其内容和功能可以分为脚本 M 文件和函数 M 文件这两大类。

（1）脚本 M 文件

它是许多 MATLAB 代码按顺序组成的命令序列集合，不接受参数的输入和输出，与 MATLAB 工作区共享变量空间。脚本 M 文件一般用来实现一个相对独立的功能，比如对某个数据集进行某种分析、绘图，求解某个已知条件下的微分方程等。用户可以通过在命令窗口直接键入文件名来运行脚本 M 文件。

通过脚本 M 文件，用户可以把为实现一个具体功能的一系列 MATLAB 代码书写在一个 M 文件中，每次只需要键入文件名即可运行脚本 M 文件中的所有代码。

（2）函数 M 文件

它也是为了实现一个单独功能的代码块，但它与脚本 M 文件不同的是需要接受参数输入和输出，函数 M 文件中的代码一般只处理输入参数传递的数据，并把处理结果作为函数输出参数返回给 MATLAB 工作区中的指定接收变量。因此，函数 M 文件具有独立的内部变量空间。在执行函数 M 文件时，要指定输入参数的实际取值，而且一般要指定接收输出结果的工作区变量。

MATLAB 提供的许多函数就是用函数 M 文件编写的，尤其是各种工具箱中的函数，用户可以打开这些 M 文件察看。实际上，对于特殊应用领域的用户，如果积累了充足的专业领域应用的函数，就可以组建自己的专业领域工具箱了。

通过函数 M 文件，用户可以把为实现一个抽象功能的 MATLAB 代码封装成一个函数接口，在以后的应用中重复调用。

### 10.1.2 MATLAB 编程概述

MATLAB 不仅是一种功能强大的高级语言，而且是一个集成的交互式开发环境，用户可以通过 MATLAB 提供的编辑调试器编写和调试 MATLAB 代码（包括脚本 M 文件和函数 M 文件）。

开发 MATLAB 程序一般需要经历代码书写、调试、优化几个阶段。MATLAB 提供了代码书写和调试的集成开发环境，用户可以在 MATLAB 的代码编辑-调试器中完成书写和调试过程。单击 MATLAB 主界面的新建工具按钮或者单击文件菜单（File）新建子菜单（New）的 M-File 项，就可以打开 MATLAB 代码编辑-调试器，其空白界面如图 10-1 所示。



图 10-1 MATLAB 代码编辑-调试器

用户也可以在命令窗口通过 edit filename 命令打开已存在的 M 文件进行编辑调试。

从图 10-1 可见，MATLAB 能够根据 M 文件内容，区别已打开的是脚本 M 文件还是函数 M 文件（如图 10-1 中，没有 function 关键字，因此被识别为脚本 M 文件），并且在整个编辑过程中追踪光标位置（如图 10-1 底部的 Ln 1 Col 1 表示当前光标处在第一行的第一列），这对于准确快速定位当前编辑和修改位置是很方便有用的。

用户在编写M文件时，要及时保存阶段性成果，可以通过File菜单的Save项，或者保存工具按钮保存当前M文件。完成代码书写之后，要试运行代码看看有没有运行错误，然后根据错误提示有针对性地对程序进行修改。

运行脚本M文件，只需要在命令窗口输入其文件名然后按回车键，或通过Debug菜单的Run项，或快捷键F5完成。运行函数M文件，需要通过命令窗口传递输入参数来调用。

除了一些很简单的代码，大部分情况下用户都可能遇到程序报错，这就需要对代码调试纠错，一般需要通过Debug菜单下的子项辅助完成，包括设置断点、单步运行等项。

当程序运行无误后，还要考虑程序性能是否可以改进。MATLAB提供了M-Lint和Profiler工具，能够辅助用户分析代码运行中时间消耗的细节和可能需要改变的编程细节，如循环赋值前没有预定义数组、用循环去实现可以用数组函数实现的运算等。这些工具都在Tools菜单下的子菜单中可以找到。

## 10.2 M文件结构和实例

### 10.2.1 M文件的一般结构

图10-2显示的是MATLAB提供的一个函数M文件的全部内容，图中清楚地显示了一般的M文件包括的各部分结构。

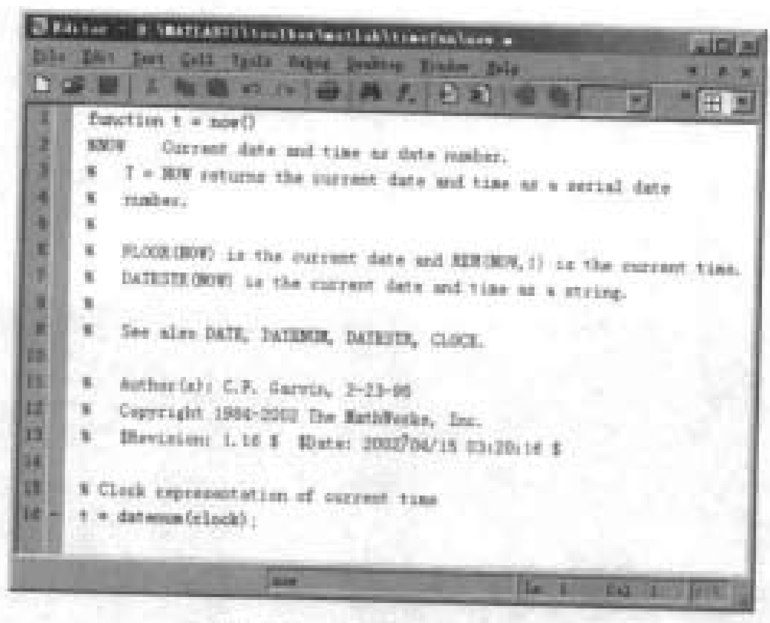



图10-2 M文件的一般结构

从图10-2可以看到，MATLAB中M文件一般包括以下五部分结构。

（1）函数声明行（Function definition line）

这一行只出现在函数M文件的第一行，通过function关键字表明此文件是一个函数M文件，并指定函数名、输入和输出参数，见图10-2中的第一行。

（2）H1 行

这是帮助文字的第一行（the first help text line），给出 M 文件帮助最关键的信息。当用 lookfor 查找某个单词相关的函数时，lookfor 只在 H1 行中搜索是否出现指定单词，见图 10-2 中的第 2 行。

（3）帮助文字

这部分对 M 文件更加详细地说明，解释 M 文件实现的功能，M 文件中出现的各变量、参数的意义，以及创作版权信息等，如图 10-2 中的第 3-13 行。当在命令窗口输入命令 help<M 文件名>时，H1 行和帮助文字部分同时显示。

（4）M 文件正文

这是 M 文件实现功能的 MATLAB 代码部分，通常包括运算、赋值等指令。图 10-2 的例子中只有第 16 行，但 M 文件的正文一般都由多行组成。

（5）注释部分

这部分出现的位置比较灵活，主要是用来注释 M 文件正文的具体运行过程，方便阅读和修改，经常穿插在 M 文件正文中间。图 10-2 的例子中的第 15 行就是注释说明正文第 16 行的意义。注释一般都是针对接下来的一段正文代码，常见的 M 文件中也经常包括多行注释。

## 10.2.2 脚本 M 文件实例

虽然一般脚本 M 文件可以包括除去函数声明行的四部分，但在实际应用中，脚本 M 文件经常仅仅由 M 文件正文和注释部分构成。正文部分实现功能，注释部分则给出每一块代码的功能说明。

**例 10-1** 脚本 M 文件实例，给出了查找 10-1000 之内所有素数的一个脚本 M 文件实例。

**解**：在命令窗口输入：

```
%Find the prime numbers between 10 and 1000
%clear the workspace
clear
%result save the prime numbers.
result=[];
%for-loop
for i=10:1000
    %define a variable which is used to mark whether i is a prime number
    mark=1;
    %check whether i is a prime number
    for j=2:i-1
        if mod(i,j)==0
            mark=0;
            break
        end
    end
    %add prime number to result
    if mark==1
```





```
            result=[result i];
        end
    end
    %output
    result
```

将例 10-1 所示编写好的文件存储为 myprime.m 文件，然后在命令窗口中键入 myprime 后回车，或者按下 F5 键或 Debug 菜单的 Run 项，都可以运行此脚本 M 文件。第一种运行方法要求 MATLAB 路径或当前目录包含 myprime.m 文件所在的目录。用户可以通过 MATLAB 主界面下 File 菜单的 Set Path…项设置 MATLAB 路径。

一般情况下，用户编写的文件都存储在 MATLAB 主目录的\work 子目录，因此按照图 10-3 所示，把\work 及其子目录都添加在 MATLAB 路径中就可以用命令行的方法访问\work 文件夹及子文件夹下的所有脚本 M 文件了。

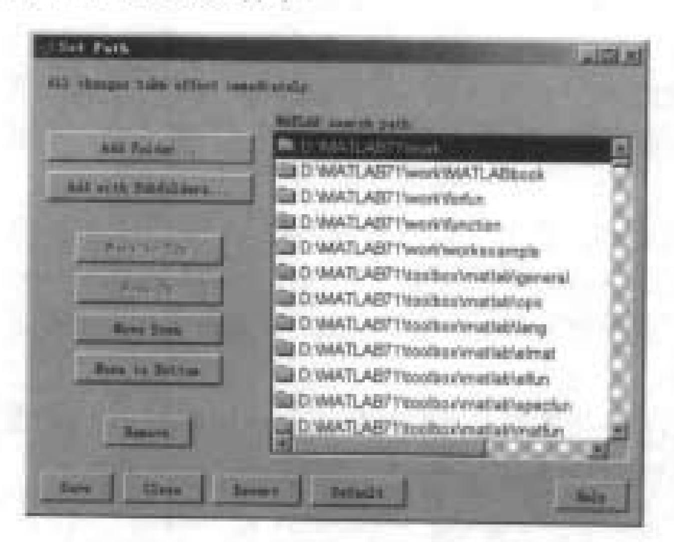



图 10-3　设置 MATLAB 路径

用户也可以通过 path 命令在 MATLAB 命令窗口下命令查看和设置 MATLAB 路径。而通过 Run 项或 F5 快捷键方式运行脚本 M 文件，则可以临时改变当前目录到此 M 文件所在的目录。

由于脚本 M 文件共享 MATLAB 工作区变量空间，因此建议用户编写脚本 M 文件时总是以 clear 开头，清除 MATLAB 当前的变量空间，防止在脚本 M 文件中出现变量调用差错。

例 10-1 的 M 脚本文件运行结果如下：

```
result =
  Columns 1 through 24
    11    13    17    19    23    29    31    37    41    43    47    53    59
61    67    71    73    79    83    89    97   101   103   107
  Columns 25 through 48
   109   113   127   131   137   139   149   151   157   163   167   173   179
181   191   193   197   199   211   223   227   229   233   239
  Columns 49 through 72
   241   251   257   263   269   271   277   281   283   293   307   311   313
317   331   337   347   349   353   359   367   373   379   383
  Columns 73 through 96
```

```
      389   397   401   409   419   421   431   433   439   443   449   457   461
463   467   479   487   491   499   503   509   521   523   541
    Columns 97 through 120
      547   557   563   569   571   577   587   593   599   601   607   613   617
619   631   641   643   647   653   659   661   673   677   683
    Columns 121 through 144
      691   701   709   719   727   733   739   743   751   757   761   769   773
787   797   809   811   821   823   827   829   839   853   857
    Columns 145 through 164
      859   863   877   881   883   887   907   911   919   929   937   941   947
953   967   971   977   983   991   997
```

### 10.2.3 函数 M 文件

一般的函数 M 文件都包括完整的五部分结构。特别强调一下，函数结构中的前三部分，函数声明行是必不可少的，用来和脚本 M 文件从结构上进行区别，并且指定函数名称和输入输出参数；H1 行简要概括函数功能，用于 lookfor 查询中；帮助文字提供函数文件的细节帮助，显示在 help 结果中，方便用户了解函数具体功能和参数意义，调用方法，以及保护作者的版权等。

函数 M 文件的命名一般习惯和函数名一致，比如图 10-2 中函数声明行 function t=now()，表明函数名为 now，因此此函数 M 文件一般保存为 now.m，调用时就可以通过 now()语句；否则，如果函数名和文件名不一致时，函数调用就需要通过文件名和与函数声明中对应的参数列表。

下面的例 10-2 给出一个作者编写的判断一个点和一个三角形位置关系的函数 M 文件。

**例 10-2** 函数 M 文件实例。

**解：**在命令窗口输入：

```
    function y=myisintri(A,M)
    %Determine whether point M is in triangle A or not.
    %A makes the triangle,A is a 3*2 matrix
    %M makes the point,M is a 1*2 vector

    %y=1 means M lies in the triangle determined by A.
    %y=0 means M lies on the edge of triangle determined by A.
    %y=-1 means M lies out of the triangle determined by A.

    %assign the coordinates of the three points
    x1=A(1,1);y1=A(1,2);
    x2=A(2,1);y2=A(2,2);
    x3=A(3,1);y3=A(3,2);
    %assigh the coordinates of the target points
    x0=M(1);y0=M(2);
    %Determine the relationship between point M and triangle A
    if
and(((M(1)-A(1,1))*(A(2,2)-A(1,2))-(A(2,1)-A(1,1))*(M(2)-A(1,2)))*((M(1)-A(1
,1))*(A(3,2)-A(1,2))-(A(3,1)-A(1,1))*(M(2)-A(1,2)))<0,((M(1)-A(2,1))*(A(1,2)
-A(2,2))-(A(1,1)-A(2,1))*(M(2)-A(2,2)))*((M(1)-A(2,1))*(A(3,2)-A(2,2))-(A(3,
1)-A(2,1))*(M(2)-A(2,2)))<0)
        y=1;
    elseif
```





```
or(((M(1)-A(1,1))*(A(2,2)-A(1,2))-(A(2,1)-A(1,1))*(M(2)-A(1,2)))*((M(1)-A(1,
1))*(A(3,2)-A(1,2))-(A(3,1)-A(1,1))*(M(2)-A(1,2)))==0,((M(1)-A(2,1))*(A(1,2)
-A(2,2))-(A(1,1)-A(2,1))*(M(2)-A(2,2)))*((M(1)-A(2,1))*(A(3,2)-A(2,2))-(A(3,
1)-A(2,1))*(M(2)-A(2,2)))==0)
      y=0;
   else
      y=-1;
   end
   %plot the triangle A and the M point
   plot([A(:,1);A(1,1)],[A(:,2);A(1,2)],'r',M(1),M(2),'b*');
```

由于此函数名为 myisintri，因此此文件最好保存为 myisintri.m。函数调用时，必须指定输入参数的值，如可以在命令窗口中用如下的指令调用 myisintri 函数：

```
>> A=rand(3,2)
A =
    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621
>> M=rand(1,2)
M =
    0.0185    0.8214
>> myisintri(A,M)    %调用后绘图结果如图 10-4 所示
ans =
    -1
```

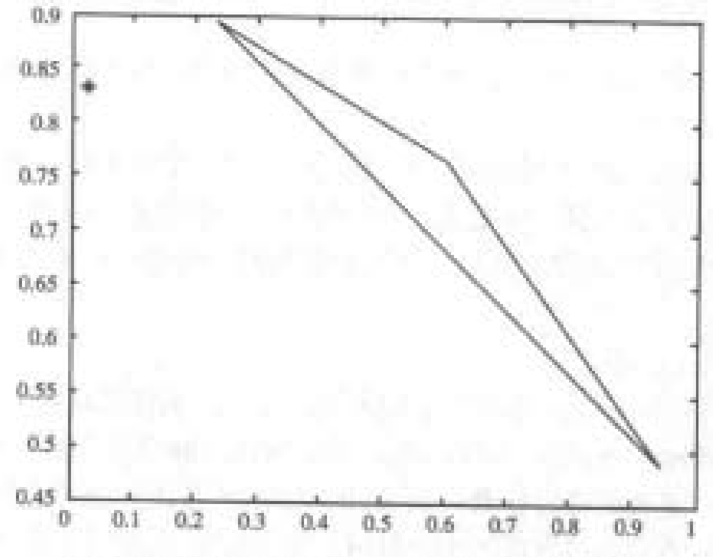



图 10-4　例 10-2 中 myisintri 函数调用后的绘图结果

编写好的函数 M 文件，相当于 MATLAB 提供的命令，可以在命令行进行函数调用。但要注意，这要求被调用的函数对应的.m 文件必须在 MATLAB 路径下，可以通过图 10-3 所示的方法设置 MATLAB 路径。

用户自己编写的脚本 M 文件和函数 M 文件，最好都保存在\work 文件夹下个人设定的各个子文件夹中，建议统一用 my 开头命名，这样对于文件和目录管理都是方便的。关于 MATLAB 文件和目录管理的内容，请参考本书第 12 章。

本章后续内容将重点讲述函数 M 文件的函数类型、参数传递和函数句柄。

## 10.3 函数类型

MATLAB 中的函数可以分为：匿名函数、M 文件主函数、嵌套函数、子函数、私有函数和重载函数。

### 10.3.1 匿名函数

匿名函数通常是很简单的函数。不像一般的 M 文件主函数要通过 M 文件编写，匿名函数是面向命令行代码的函数形式，它通常只由一句很简单的声明语句组成。

与一般 M 文件函数一样的是，匿名函数也可以接受多个输入和输出参数。使用匿名函数的优点是不需要维护一个 M 文件，而只需要一句非常简单的语句，就可以在命令窗口或 M 文件中调用函数，这对于那些函数内容非常简单的情况是很方便的。

创建匿名函数的标准格式是：

fhandle = @(arglist) expr

其中，

（1）expr 是通常是一个简单的 MATLAB 变量表达式，实现函数的功能，比如 *x*+*x*.^2，sin(*x*).*cos(*x*)等；

（2）arglist 是参数列表，它指定函数的输入参数列表，对于多个输入参数的情况，通常要用逗号分隔各个参数；

（3）符号@是 MATLAB 中创建函数句柄的操作符，表示创建由输入参数列表 arglist 和表达式 expr 确定的函数句柄，并把这个函数句柄返回给变量 fhandle，这样，以后就可以通过 fhandle 来调用定义好的这个函数。关于函数句柄的内容，请参考本章 10.6 节。

例如定义函数：

myfunhd=@(*x*)(*x*+*x*.^2)

表示创建了一个匿名函数，它有一个输入参数 *x*，它实现的功能是 *x*+*x*.^2，并把这个函数句柄保存在变量 myfunhd 中，以后就可以通过 myfunhd(*a*)来计算当 *x* = *a* 时的函数值。

需要注意的是，匿名函数的参数列表 arglist 中可以包含一个参数或多个参数，这样调用的时候就要按顺序给出这些参数的实际取值。但 arglist 也可以不包含参数，即留空，这种情况下调用函数时还是需要通过 fhandle()的形式来调用，即要在函数句柄后紧跟一个空的括号，否则，只显示 fhandle 句柄对应的函数形式。

匿名函数可以嵌套，即在 expr 表达式中可以用函数来调用一个匿名函数句柄。

**例 10-3** 匿名函数。

**解**：在命令窗口输入：

```
>> myfhd1=@(x)(x+x.^2)
myfhd1 =
    @(x)(x+x.^2)
>> myfhd1(2)
```





```
ans =
     6
>> myfhd2=@(x,y)(sin(x)+cos(y))
myfhd2 =
    @(x,y)(sin(x)+cos(y))
>> myfhd2(pi/2,pi/6)
ans =
    1.8660
>> myfhd3=@()(3+2)
myfhd3 =
    @()(3+2)
>> myfhd3()
ans =
     5
>> myfhd3
myfhd3 =
    @()(3+2)
>> myffhd=@(a)(quad(@(x)(a.*x.^2+1./a.*x+1./a^2),0,1)) %匿名函数嵌套使用
myffhd =
    @(a)(quad(@(x)(a.*x.^2+1./a.*x+1./a^2),0,1))
>> myffhd(0.5)
ans =
    5.1667
```

匿名函数可以保存在.mat 文件中，例 10-3 中就可以通过 save myfunquad.mat myffhd 把匿名函数句柄 myffhd 保存在 myfunquad.mat 文件中，以后需要用到匿名函数 myffhd 时，只需要 load myfunquad.mat myffhd 就可以。

### 10.3.2　M 文件主函数

每一个函数 M 文件第一行定义的函数就是 M 文件主函数，一个 M 文件只能包含一个主函数，并通常习惯上将 M 文件名和 M 文件主函数名设为一致的。

M 文件主函数的说法是针对其内部嵌套函数和子函数而言的。一个 M 文件中除了一个主函数外，还可以编写多个嵌套函数或子函数，以便在主函数功能实现中进行调用。

### 10.3.3　嵌套函数

在一个函数内部，可以定义一个或多个函数，这种定义在其他函数内部的函数就被称为嵌套函数。嵌套可以多层发生，就是说一个函数内部可以嵌套多个函数，这些嵌套函数内部又可以继续嵌套其他函数。

嵌套函数的书写语法格式是：

```
function x = A(p1, p2)
...
   function y = B(p3)
   ...
   end
...
end
```

一般函数代码中结尾是不需要专门标明 end 的，但在使用嵌套函数时，无论是嵌套函数还是嵌套函数的父函数（直接上一层次的函数）都要明确标出 end 表示函数结束。

嵌套函数的互相调用需要注意，和嵌套的层次密切相关。如下面一段代码中：

（1）外层的函数可以调用向内一层直接嵌套的函数（***A*** 可以调用 ***B*** 和 ***D***），而不能调用更深层的嵌套函数（***A*** 不可以调用 ***C*** 或 ***E***）；

（2）嵌套函数可以调用与自己具有相同父函数的其他同层嵌套函数（***B*** 和 ***D*** 可以互相调用）；

（3）嵌套函数也可以调用其父函数，或与父函数具有相同父函数的其他嵌套函数（***C*** 可以调用 ***B*** 和 ***D***），但不能调用与其父函数具有相同父函数的其他嵌套函数内深层嵌套的函数。

```
function A(x, y)    %外层函数 A（例如主函数）
B(x, y);
D(y);

   function B(x, y) %A 的嵌套函数（以 A 为参照可以称为第一层嵌套函数），B 的父函数为 A
   C(x);
   D(y);

      function C(x) %B 的嵌套函数（以 A 为参照可以称为第二层嵌套函数），C 的父函数为 B
      D(x);
      end
   end

   function D(x)    %A 的嵌套函数（以 A 为参照可以称为第一层嵌套函数），D 的父函数为 A
   E(x);

      function E(x) %D 的嵌套函数（以 A 为参照可以称为第二层嵌套函数），F 的父函数为 D
      ...
      end
   end
end
```

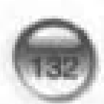
## 10.3.4 子函数

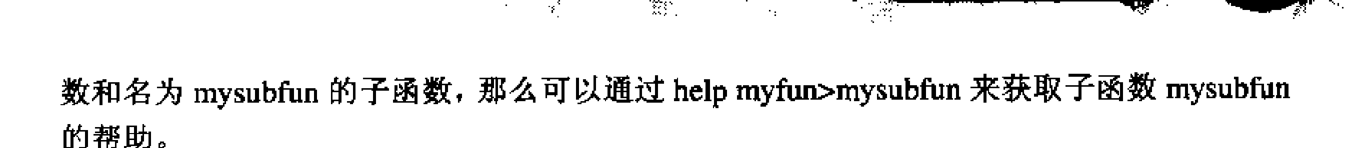
一个 M 文件只能包含一个主函数，但一个 M 文件中可以包含多个函数，这些编写在主函数后的函数都称为子函数。所有子函数只能被其所在 M 文件中的主函数或其他子函数调用。

所有子函数都有自己独立的声明和帮助、注释等结构，只需要在位置上处在主函数之后即可，而各个子函数的前后顺序都可以任意放置，和被调用的前后顺序无关。

M 文件内部发生函数调用时，MATLAB 首先检查该 M 文件中是否存在相应名称的子函数，然后检查这一 M 文件所在目录的子目录下是否存在同名的私有函数，然后按照 MATLAB 路径，检查是否存在同名的 M 文件或内部函数。根据这一顺序，函数调用时首先查找相应的子函数，因此，可以通过编写同名子函数的方法实现 M 文件内部的函数重载。

子函数的帮助文件也可以通过 help 命令显示，如 myfun.m 文件中有名为 myfun 的主函





数和名为 mysubfun 的子函数，那么可以通过 help myfun>mysubfun 来获取子函数 mysubfun 的帮助。

### 10.3.5 私有函数

私有函数是具有限制性访问权限的函数，它们对应的 M 文件需要保存在名为 private 的文件夹下，这些私有函数代码编写上和普通的函数没有什么区别，也可以在一个 M 文件中编写一个主函数和多个子函数，以及嵌套函数。但私有函数只能被 private 目录的直接父目录下的脚本 M 文件或 M 文件主函数调用。

通过 help 命令获取私有函数的帮助，也需要声明其私有特点，例如要获取私有函数 myprifun 的帮助，就要通过 help private/myprifun 命令。

### 10.3.6 重载函数

重载是计算机编程中非常重要的概念，它经常的用在处理功能类似，但参数类型或个数不同的函数编写中。例如现在要实现一个计算功能，一种情况下输入的几个参数都是双精度浮点类型，同时也有一种情况是输入的几个参数都是整型变量，这时候，用户就可以编写两个同名函数，一个用来处理双精度浮点类型的输入参数，另一个用来处理整型的输入参数，这样，当用户实际调用函数时，MATLAB 就会根据实际传递的变量类型选择执行其中一个函数。

MATLAB 中重载函数通常放置在不同的文件夹下，通常文件夹名称以符号@开头，然后跟一个代表 MATLAB 数据类型的字符，如@double 目录下的重载函数的输入参数应该是双精度浮点型，而@int32 目录下的重载函数的输入参数应该是 32 位整型。

## 10.4 参数传递

### 10.4.1 MATLAB 参数传递概述

MATLAB 中通过 M 文件编写函数时，只需要指定输入和输出的形式参数列表，只是在函数实际被调用的时候，才需要把具体的数值提供给函数声明中给出的输入参数。

MATLAB 中参数传递过程是传值传递，也就是说，在函数调用过程中，MATLAB 将传入的实际变量值赋给形式参数指定的变量名，这些变量都存储在函数的变量空间中，这和工作区变量空间是独立的，每一个函数在调用中都有自己独立的函数空间。

例如编写函数：

function y=myfun(*x*,*y*,*z*)

在命令窗口通过 *a*=myfun(3,2,0.5)调用此函数，那么 MATLAB 首先会建立 myfun 函数的变量空间，把 3 赋值给 *x*，把 2 赋值给 *y*，把 0.5 赋值给 *z*，然后执行函数实现的代码，



在执行完毕后，把 myfun 函数返回的参数 y 的值传递给工作区变量 $a$，调用过程结束后，函数变量空间被清除。

### 10.4.2 输入和输出参数的数目

MATLAB 的函数可以具有多个输入或输出参数。通常在调用时，需要给出和函数声明语句中一一对应的输入参数；而输出参数个数可以按参数列表对应指定，也可以不指定。不指定输出参数调用函数时，MATLAB 默认地把输出参数列表中的第一个参数的值返回给工作区变量 ans。

MATLAB 中可以通过 nargin 和 nargout 函数，确定函数调用时实际传递的输入和输出参数个数，结合条件分支语句，就可以处理函数调用中指定不同数目的输入输出参数的情况。

**例 10-4** 输入和输出参数的数目。

**解**：在命令窗口输入：

```
function [y1,y2]=mytestnio(x1,x2)
if nargin==1
    y1=x1;
    if nargout==2
        y2=x1;
    end
else
    if nargout==1
        y1=x1+x2;
    else
        y1=x1;
        y2=x2;
    end
end
```

这个函数可以处理 1 或 2 个输入参数，1 或 2 个输出参数的情况。当只有 1 个输入参数 $x1$ 和 1 个输出参数 y1 时，把 $x1$ 赋值给 y1；当有 1 个输入参数 $x1$ 和 2 个输出参数 y1，y2 时，把 $x1$ 赋值给 y1 和 y2；当有 2 个输入参数 $x1$，$x2$ 和 1 个输出参数 y1 时，把 $x1+x2$ 的计算结果赋值给 y1；当有 2 个输入参数 $x1$，$x2$ 和 2 个输出参数 y1，y2 时，把 $x1$ 赋值给 y1，把 $x2$ 赋值给 y2。函数调用结果如下：

```
>> x=mytestnio(5)
x =
     5
>> [x,y]=mytestnio(5)
x =
     5
y =
     5
>> mytestnio(5)
ans =
     5
>> x=mytestnio(5,7)
```



```
x =
    12
>> [x,y]=mytestnio(5,7)
x =
     5
y =
     7
>> mytestnio(5,7)
ans =
     5
```

指定了输入和输出参数个数的情况比较好理解，只要对应函数 M 文件中对应的 if 分支项即可；而不指定输出参数个数的调用情况，MATLAB 是按照指定了所有输出参数的调用格式对函数进行调用的，不过在输出时只是把第一个输出参数对应的变量值赋给工作区变量 ans。如 mytestnio(5,7)这句函数调用中，实际上是按照[y1,y2]=mytestnio(*x*1,*x*2)这种形式调用的，在函数变量空间中 *x*1 被赋值为 5，*x*2 被赋值为 7，y1 计算结果为 5，y2 计算结果为 7，但函数只把输出参数列表中的第一个输出变量（即 y1）的取值返回给工作区变量 ans，因此，ans 取值为 5。

### 10.4.3 可变数目的参数传递

函数 nargin 和 nargout 结合条件分支语句，可以处理可能具有不同数目的输入和输出参数的函数调用，但这要求对每一种输入参数数目和输出参数数目的组合分别进行代码编写。有些情况下，用户可能并不能确定具体调用中传递的输入参数或输出参数的个数，即具有可变数目的传递参数，MATLAB 中可以通过 varargin 和 varargout 函数实现可变数目的参数传递，使用这两个函数对于处理具有复杂的输入输出参数个数组合的情况也是便利的。

函数 varargin 和 varargout 把实际的函数调用时传递的参数值封装成一个元胞数组，因此，在函数实现部分的代码编写中，就要用访问元胞数组的方法访问封装在 varargin 或 varargout 中的元胞或元胞内的变量。

**例 10-5** 可变数目的参数传递。

**解：** 在命令窗口输入：

```
function y=mytestvario(varargin)
temp=0;
for i=1:length(varargin)
   temp=temp+mean(varargin{i}(:));
end
y=temp/length(varargin);
```

例 10-5 的函数 mytestvario 以 varargin 为输入参数，从而可以接受可变数目的输入参数。函数实现部分首先计算了各个输入参数（可能是标量、一维数组或二维数组）的均值，然后计算这些均值的均值。调用结果如下：

```
>> mytestvario(4)
ans =
     4
>> mytestvario(4,[1 3])
```

```
ans =
    3
>> mytestvario(4,[1 3],[1 23;23 1],magic(4))
ans =
   6.6250
```

对于 mytestvario(4,[1 3],[1 23;23 1],magic(4))这句函数调用，在函数变量区，varargin 首先被赋值为一个元胞数组{4,[1 3],[1 23;23 1],magic(4)}，即 varargin 有 1 行 4 列个元胞，各个元胞中分别存储了一个标量数值、一维行数组、2 行 2 列的二维数组和 4 行 4 列的魔方数组。在函数实现部分，首先创建中间变量 temp，并初始化赋值为 0（用来存储各个元胞中数据均值的总和），然后计算每一个元胞中所有数据的均值并将结果累加到 temp 上，最后通过 y=temp/length(varargin)计算这些均值的均值。

函数 varargin 和 varargout 也可以在参数列表中，放置在某些必然出现的参数之后，其语法格式可以如下：

（1）function [out1,out2] = example1(*a*,*b*,varargin)

表示函数 example1 可以接受大于等于 2 个输入参数，返回两个输出参数，2 个必选的输入参数是 *a* 和 *b*，其他更多的输入参数被封装在 varargin 中。

（2）function [*i*,*j*,varargout] = example2(*x*,*y*)

表示函数 example2 接受 2 个输入参数 *x* 和 *y*，返回大于等于 2 个输出参数，前两个输出参数为 *i* 和 *j*，其他更多的输出参数封装在 varargin 中。

函数 varargout 和 varargin 的用法类似，就只需要注意访问时要按照访问元胞数组的方法，这里就不再举例了。

## 10.4.4 返回被修改的输入参数

前面已经讲过，MATLAB 函数有独立于 MATLAB 工作区的自己的变量空间，因此，输入参数在函数内部的修改，都只具有和函数变量空间相同的生命期，如果不指定将此修改后的输入参数值返回到工作区空间，那么在函数调用结束后，这些修改后的值将被自动清除。

**例 10-6** 函数内部的输入参数修改。

**解：**在命令窗口输入：

```
function y=mytest(x)
x=x+5;
y=x*2;
```

例 10-6 中的 mytest 函数内部，首先修改了输入参数 *x* 的值（*x*=*x*+5），然后以修改后的 *x* 的值计算输出参数 y 的值（*y*=*x**2）。调用结果如下：

```
>> x=3
x =
    3
>> y=mytest(x)
y =
   16
>> x
```





```
x =
     3
```

由此结果可见，调用结束后，函数变量区中的 $x$ 在函数调用中被修改，但此修改只在函数变量区有效，这并没有影响到 MATLAB 工作区变量空间中的变量 $x$ 的值，函数调用前后，MATLAB 工作区中的变量 $x$ 始终取值为 3。

那么，如果用户希望函数内部对输入参数的修改也对 MATLAB 工作区的变量有效，那么就需要在函数输出参数列表中返回此输入参数。对例 10-6 的函数，则需要把函数修改为 function [y,*x*]=mytest(*x*)，而在调用时也要通过[y,*x*]= mytest(*x*)语句。

**例 10-7** 将修改后的输入参数返回给 MATLAB 工作区。

**解：**在命令窗口输入：

```
function [y,x]=mynewtest(x)
x=x+5;
y=x*2;
```

MATLAB 工作区中的调用结果如下：

```
>> x=3
x =
     3
>> [y,x]=mynewtest(x)
y =
    16
x =
     8
>> x
x =
     8
```

通过例 10-7 可见，函数调用后，MATLAB 工作区中的变量 x 取值从 3 变为 8（3+5），可见通过[y,*x*]=mynewtest(*x*)调用，实现了函数对 MATLAB 工作区变量的修改。

### 10.4.5 全局变量

通过返回修改后的输入参数，可以实现函数内部对 MATLAB 工作区变量的修改。而另一种殊途同归的方法则是使用全局变量。声明全局变量需要用到 global 关键词，语法格式为 global variable。

通过全局变量可以实现 MATLAB 工作区变量空间和多个函数的函数空间的共享，这样，多个使用全局变量的函数和 MATLAB 工作区共同维护这一全局变量，任何一处对全局变量的修改，都会直接改变此全局变量的取值。

在应用全局变量时，通常在各个函数内部通过 global variable 语句声明，在命令窗口或脚本 M 文件中也要先通过 global 声明，然后进行赋值和调用。

**例 10-8** 全局变量的使用。

**解：**在命令窗口输入：

```
function y=myprocess(x)
```

```
global T
T=T*2;
y=exp(T)*sin(x);
```

然后在命令窗口声明全局变量然后赋值调用：

```
>> global T
>> T=0.3
T =
    0.3000
>> myprocess(pi/2)
ans =
    1.8221
>> exp(T)*sin(pi/2)
ans =
    1.8221
>> T
T =
    0.6000
```

通过例 10-8 可见，用 global 将 *T* 声明为全局变量后，函数内部对 *T* 的修改也会直接作用到 MATLAB 工作区中。函数 myprocess 调用一次后，*T* 的值从 0.3 变为 0.6（0.3*2）。

## 10.5 函数句柄

### 10.5.1 函数句柄的创建和调用

函数句柄实际上是提供了一种间接调用函数的方法。创建函数句柄需要用到操作符@。前面已经讲过，匿名函数实际上就是一种函数句柄，而对 MATLAB 提供的各种 M 文件函数和内部函数，也都可以创建函数句柄，从而可以通过函数句柄对这些函数实现间接调用。

创建函数句柄的一般语法格式为：

fhandle=@function_fileame

其中：

（1）funciont_filename 是函数所对应的 M 文件的名称或 MATLAB 内部函数的名称；

（2）@是句柄创建操作符；

（3）fhandle 变量保存这一函数句柄。

例如 fhandle=@sin 就创建了 MATLAB 内部函数 sin 的句柄，并将之保存在 fhandle 变量中，以后就可以通过 fhandle(*x*)来实现 sin(*x*)的功能。

通过函数句柄调用函数时，也需要指定函数的输入参数，比如可以通过 fhandle(arg1, arg2,⋯, arg*n*)这样的调用格式来调用具有多个输入参数的函数，对于那些没有输入参数的函数，在使用句柄调用时，要在句柄变量后加上空的圆括号，即 fhandle()。

**例 10-9** 函数句柄的创建和调用。

**解：**在命令窗口输入：

```
>> fhd=@sin
fhd =
```





```
    @sin
>> x=0:0.25*pi:2*pi;
>> fhd(x)
ans =
         0    0.7071    1.0000    0.7071    0.0000   -0.7071   -1.0000
-0.7071   -0.0000
```

### 10.5.2 处理函数句柄的函数

MATLAB 中提供了丰富的处理函数句柄的函数，如表 10-1 所示。

表 10-1　处理函数句柄的函数

| 函　数 | 说　明 |
|---|---|
| functions(fhandle) | 返回一个结构体，存储了函数的名称，函数类型（简单函数或重载函数），以及函数 M 文件的位置 |
| func2str(fhandle) | 将函数句柄转换为函数名称字符串 |
| str2func(str) | 将字符串代表的函数转换为函数句柄 |
| save filename.mat fhandle | 将函数句柄保存在.mat 文件中 |
| load filename.mat fhandle | 把.mat 文件中存储的函数句柄装载到工作区 |
| isa(var, 'function_handle') | 检测变量 var 是不是函数句柄 |
| isequal(fhda,fhdb) | 检测两个函数句柄是否对应于同一个函数 |

**例 10-10** 处理函数句柄的函数。

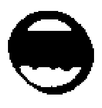
**解**：在命令窗口输入：

```
>> fhda=@exp
fhda =
    @exp
>> fhdb=@myprocess
fhdb =
    @myprocess
>> functions(fhdb)
ans =
    function: 'myprocess'
        type: 'simple'
        file: 'D:\MATLAB71\work\MATLABbook\EX-10\myprocess.m'
>> isa(fhda,'function_handle')
ans =
     1
>> isequal(fhda,fhdb)
ans =
     0
```

## 10.6 小结

本章从 MATLAB 中 M 文件编程入手，顺序介绍了 M 文件结构、MATLAB 中的各种函数类型、函数参数传递以及函数句柄的应用等内容。

通过本章学习，读者应该了解到开发 MATLAB 程序的一般流程，理解脚本 M 文件和函数 M 文件在结构、功能、应用范围上的差别，熟悉并掌握 MATLAB 中各种类型的函数，尤其对匿名函数，以 M 文件为核心的 M 文件主函数、子函数、嵌套函数要能熟练应用，对于函数句柄的创建和调用也要理解和掌握。对于中高级 M 函数编程用户，还要熟悉参数传递过程及相关函数，以及句柄处理函数。



# 第 11 章

# M 文件调试和剖析

第 10 章介绍了 MATLAB 中通过 M 文件编写脚本或函数代码的知识，本章内容紧接前一章，讲述调试和分析 M 文件的工具和方法。

## 11.1　M 文件调试工具

当完成 MATLAB 代码编写后，用户就可以在命令窗口运行代码（脚本或函数文件）。对于比较简单的代码，一般只要编程习惯较好，都可以一次通过，但对于很多比较复杂的情况，或者用户初学 MATLAB 编程，一些常见的错误还不能避免，容易在运行时出现错误，这时候，就需要利用 MATLAB 的调试工具对出现错误的代码进行调试纠错。

MATLAB 的代码编辑调试器是一个综合了代码编写、调试的集成开发环境。MATLAB 代码调试过程，主要是通过 MATLAB 代码编辑器的 Debug 菜单下的子项，如图 11-1 所示。

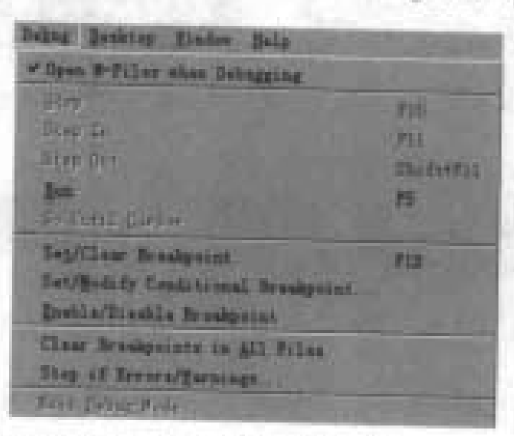



图 11-1　MATLAB 代码编辑调试器的 Debug 菜单

Debug 菜单下各子项的含义如下。

（1）Step：在调试模式下，执行 M 文件的当前行，对应的快捷键是 F10。

（2）Step In：在调试模式下，执行 M 文件的当前行，如果 M 文件当前行调用了另一个函数，那么进入该函数内部，对应的快捷键是 F11。

（3）Step Out：当在调试模式下执行 Step In 进入某个函数内部之后，执行 Step Out 可以完成函数剩余部分的所有代码，并退出函数，暂停在进入函数内部前的 M 文件所在行末尾。

（4）Run：运行当前 M 文件，快捷键是 F5；当前 M 文件设置了断点时，运行到断点处暂停。

（5）Go Until Cursor：运行当前 M 文件到在光标所在行尾。

需要注意，以上这些调试项，除了 Run，都需要首先在 M 文件中设置断点，然后 Run 运行到断点位置后，这些调试项才可启用。

（6）Set/Clear Breakpoint：在光标所在行开头设置或清除断点。

（7）Set/Modify Conditional Breakpoint…：在光标所在行开头设置或修改条件断点，选择此子项，会打开条件断点设置对话框如图 11-2 所示，用于设置在满足什么条件时，此处断点有效。

（8）Enable/Disable Breakpoint：将当前行的断点设置为有效或无效。

（9）Clear Breakpoints in All Files：清除所有 M 文件中的断点。

（10）Stop if Errors/Warnings…：设置出现某种运行错误或警告时，停止程序运行，选择此子项，会打开错误/警告设置对话框，如图 11-3 所示。

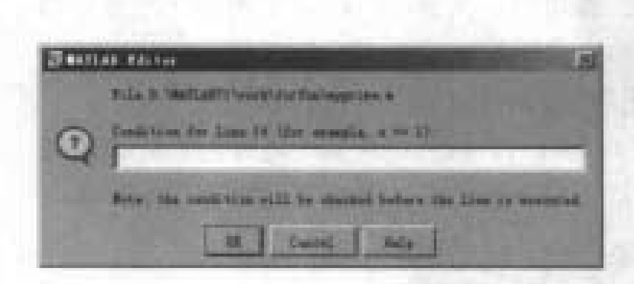


图 11-2　条件断点设置对话框

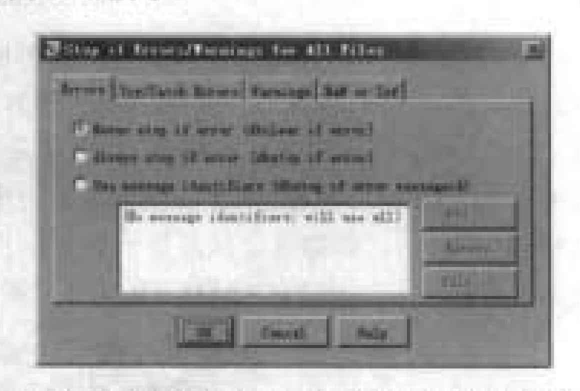


图 11-3　设置出现某种运行错误或警告则停止程序运行

（11）Exit Debug Mode：退出调试模式。

上面逐项讲述了 Debug 菜单下每一个子项的意义，实际上，很多子项都有对应的快捷工具按钮。MATLAB 代码编辑调试器中，如图 11-4 所示的部分工具按钮就是用于 M 文件调试的。



图 11-4　调试工具按钮

图 11-4 中的各个工具按钮，从左向右依次对应于 Set/Clear Breakpoint，Clear Breakpoints in All Files，Step，Step In，Step Out，Run，Exit Debug Mode 等菜单子项。





通常的调试过程是：先单击 Run 按钮，运行一遍 M 文件，针对系统给出的具体的出错信息，在适当的地方设置断点或条件断点，再次运行到断点位置（如图 11-5 所示），此时 MATLAB 把运行控制权交给键盘，命令窗口出现“K>>”提示符（如图 11-6 所示），此时可以在命令窗口查询 M 文件运行过程中的所有变量，包括函数运行时的中间变量。运行到断点位置后，用户可以选择 Step/Step Into/Step Out 等调试运行方式，逐行运行并适时查询变量取值，从而逐渐找到错误所在并将其排除。

图 11-5　设置断点后 Run 运行到断点所在位置

图 11-6　调试模式时 MATLAB 命令窗口把控制权交给键盘

## 11.2　M 文件分析工具

通过 Debug 项对 M 文件进行调试，可以对文件中的编写错误和运行错误进行纠正。完成了调试后，用户编写的 M 文件就可以正确的运行了，但可能运行效率还不是最优，这就

需要通过 MATLAB 提供的分析工具对代码进行分析，然后有针对性地进行优化。

MATLAB 提供的 M 文件分析工具包括 M-Lint 工具和 Profiler 工具，它们都有图形操作界面，使用简单方便，是 MATLAB 程序分析优化的必用工具。

### 11.2.1 M-Lint 分析工具

M-Lint 工具可以分析用户 M 文件中的错误或性能问题。用户可以先在代码编辑调试器中打开待分析的 M 文件，然后选择 Tools 菜单下的 Check Code with M-Lint 项，如图 11-7 所示。

图 11-7　通过 Tools 菜单打开 M-Lint 工具

图 11-7 中的代码是本书第 10 章例 10-1 的 M 文件。运行 M-Lint 工具后结果如图 11-8 所示。

从图 11-8 可以看出，M-Lint 分析完成后，会返回一个浏览器界面下的分析报告（Check Report），报告中包括被分析的 M 文件的路径，以及若干个分析结果（图 11-8 所示的 1 message 表示只有一条分析结果）。分析结果的格式是“行号：错误或问题报告”。

M-Lint 分析结果中经常出现的错误或问题报告包括：没有以分号结束以阻止中间变量输出、变量在文件中从没有被其他语句调用、循环过程中数组尺寸会增加等。



实际上，M-Lint 分析得到的问题报告，并不一定必须要消除，要具体问题具体分析。当用户认可某一条分析结果时，可以点击分析结果中的行号，就可以快捷打开相应的 M 文件并定位到该行，用户就可以方便地修改代码了。

M-Lint 不仅可以分析单个 M 文件，还可以分析一个文件夹下的所有 M 文件。通常在 MATLAB 主界面下，选择 Desktop 菜单下的 Current Directory，则可以显示文件夹面板，通过单击此面板顶部的 M-Lint 工具则可以分析相应文件夹下所有 M 文件，如图 11-9 所示。

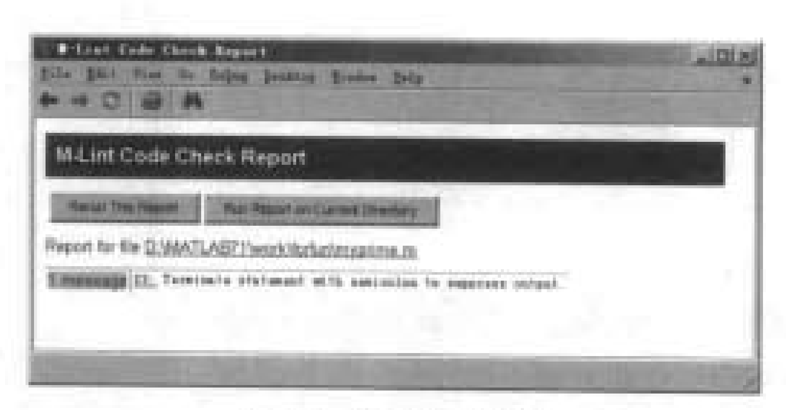



图 11-8　M-Lint 分析结果

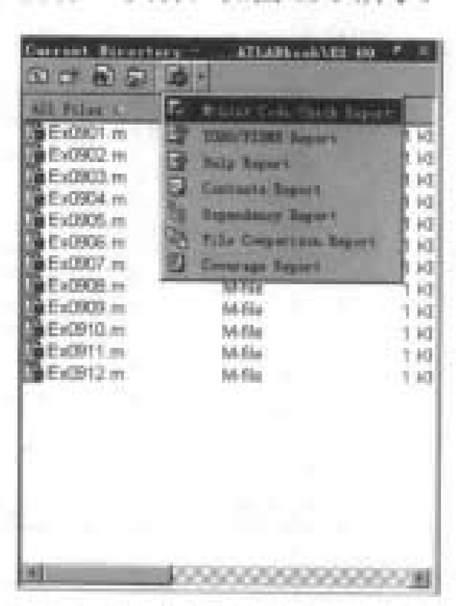


图 11-9　当前目录面板和 M-Lint 按钮

## 11.2.2　Profiler 分析工具

Profiler 工具是 MATLAB 提供的另一个功能强大的代码分析工具。使用时，用户可以提前在代码编辑调试器中打开 M 文件，然后选择 Tools 菜单的 Open Profiler 项，就可以运行 Profiler 分析工具，Profiler 图形界面如图 11-10 所示。

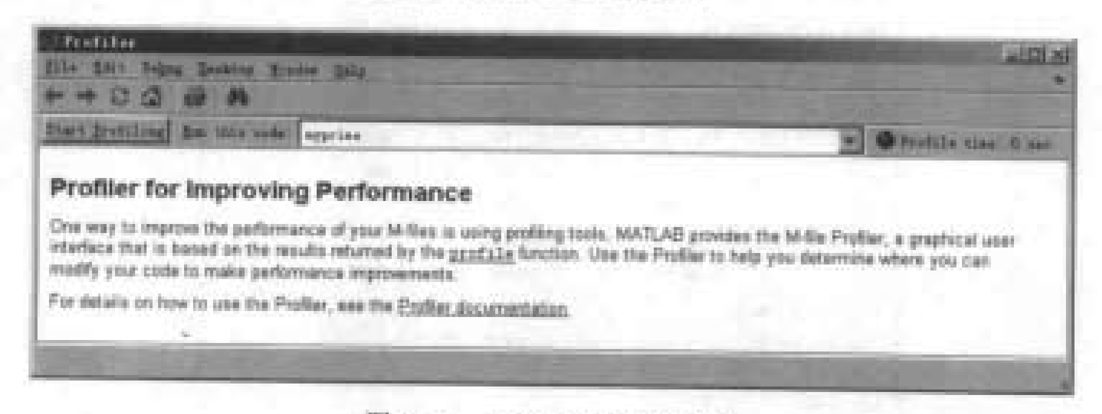



图 11-10　Profiler 工具图形界面

单击图 11-10 中的 Start Profiling 按钮，就可以分析此 M 文件，分析结果如图 11-11 所示。

从图 11-11 中可见，Profiler 分析结果给出了调用函数名称、调用次数、消耗总时间等信息，单击图 11-11 中蓝色的 myprime，可以打开更加详细的关于此 M 文件的分析报告，如图 11-12 所示。

图 11-12 更详细的 Profiler 分析结果中显示了 myprime.m 文件运行中最消耗时间的部分及其具体耗时信息。用户可以有针对性地修改那些最消耗时间的部分。

一般来说，应该尽量避免不必要的变量输出，循环赋值前预定义数组尺寸，多采用向量化的 MATLAB 函数，少采用数组，这些都能够提高 MATLAB 程序的运行性能。

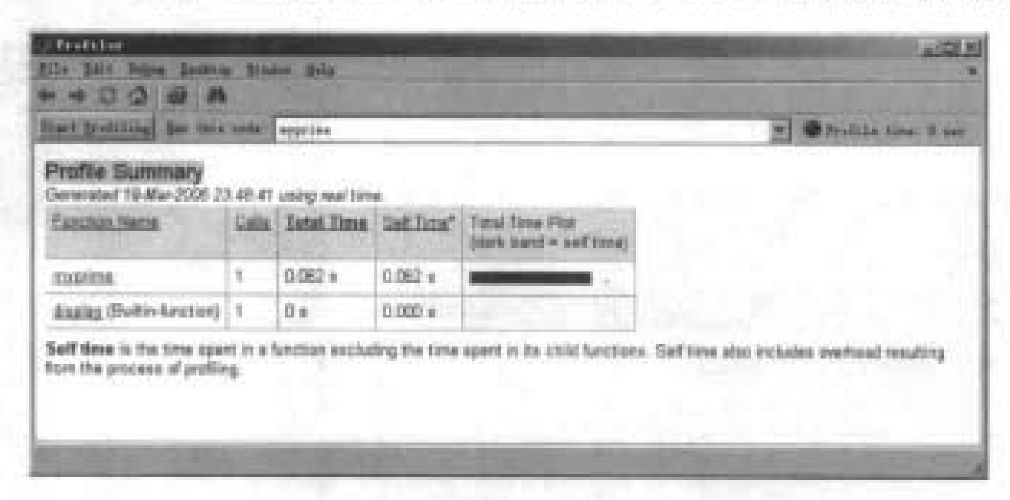



图 11-11 Profiler 分析结果

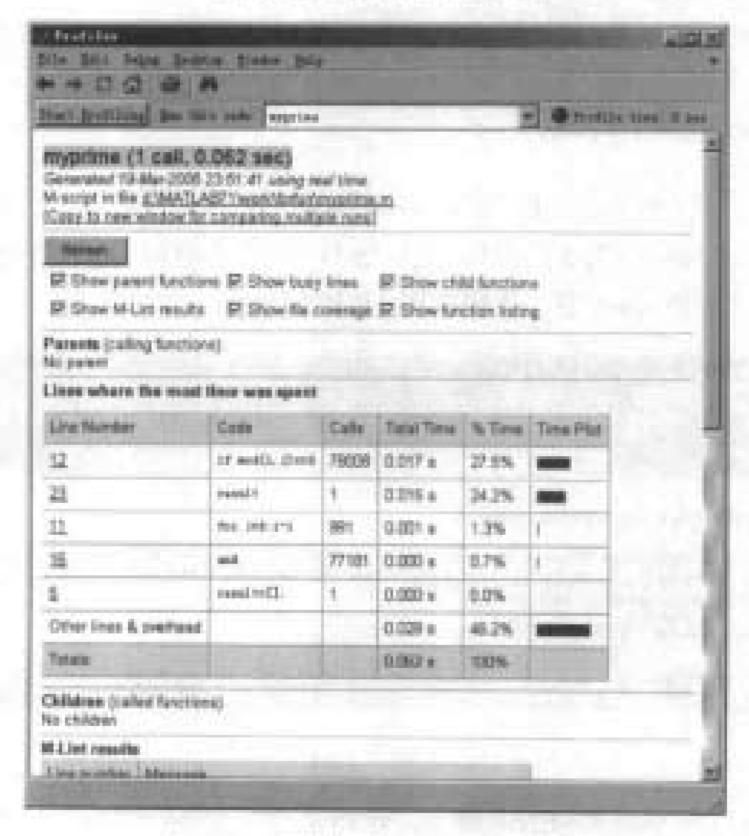



图 11-12 更详细的 Profiler 分析结果

## 11.3 小结

本章着重讲述了 MATLAB 中的代码调试和代码分析优化过程及相应的工具，这部分内容对于中高级 MATLAB 编程用户是应该熟练掌握的，而本章只是浅显地概述了这些工具、过程，一般的 MATLAB 用户都应该有所了解。尤其是断点调试部分的内容，建议读者尽量以自己的程序代码为例，多加练习，熟练掌握。



# 第 12 章

# 目录管理和文件 I/O

MATLAB 所有文件操作（包括 M 文件运行、创建等）都以当前目录和 MATLAB 搜索路径为参考。因此，灵活切换当前目录位置和设置 MATLAB 搜索路径，是运行脚本或函数 M 文件的重要前提。

另一方面，MATLAB 编程用户，以及使用 MATLAB 中面临大规模数据 I/O 的用户，在 MATLAB 工作空间和本地磁盘文件之间的数据导入导出方面会有特别的需要。

本章就针对性地简单讲述 MATLAB 对这两类问题的解决方法：目录管理和搜索路径设置、数据导入导出的文件 I/O。

## 12.1 当前目录和目录管理

### 12.1.1 当前目录工具条

用户运行个人编写的 MATLAB 脚本 M 文件或函数 M 文件时，一个简单的处理方法是把这些文件放置在当前目录下，或将当前目录改变到文件所在的目录。

MATLAB 主界面的工具条中就能显示和设置当前目录，如图 12-1 所示。图中显示当前目录为：d:\MATLAB71 \work。通过图 12-1 所示的工具条，用户可以很方便地变更当前目录的位置。单击目录文本框使目录变为选中状态，用户就可以手动输入一个新目录来修改当前目录位置了。

图 12-1　当前目录显示和设置工具条

单击文本框右边的下拉三角形，会出现最近几次设置的当前目录，如图 12-2 所示，用户可以选择这些目录来快速改变当前目录。



单击紧邻的右边的目录浏览按钮，会弹出一个浏览文件夹窗口（如图 12-3 所示），允许用户选择某个本地目录作为当前目录。

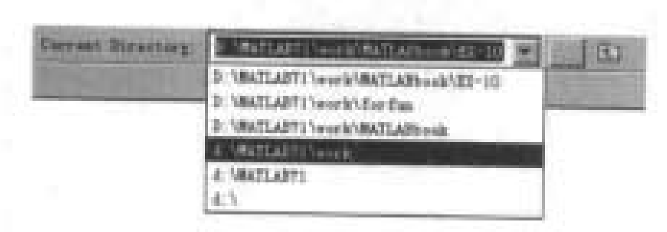


图 12-2　通过下拉三角形设置切换当前目录



图 12-3　浏览文件夹窗口

图 12-1 中最右边的按钮的功能是把当前目录切换到父目录，即将当前目录上移一层，例如对图 12-1 所示的当前目录，上移一层就会切换到 d:\MATLAB71。

## 12.1.2　当前目录面板

MATLAB 中提供了当前目录面板，可以供用户浏览当前目录下的文件和文件夹的各种属性。默认界面下，当前目录界面是不显示的，打开当前目录界面的方法是：选择 Desktop 菜单的 Current Directory 项，当前目录面板如图 12-4 所示。

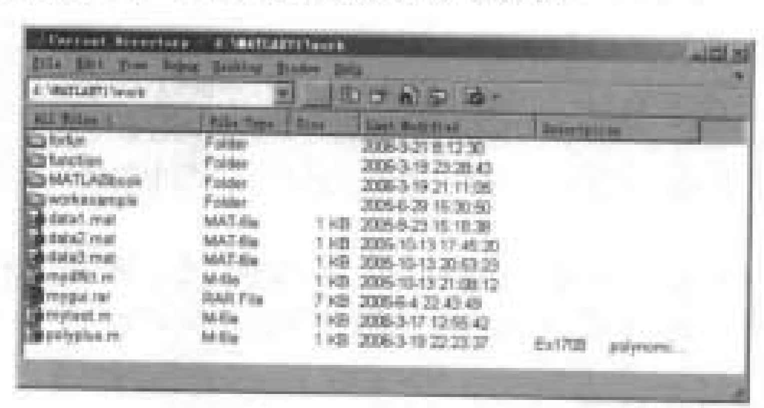


图 12-4　当前目录面板

图 12-4 是当前目录面板单独显示的界面，单击菜单栏最右侧的箭头标志，可以将当前窗口附着显示在 MATLAB 主界面侧边栏。

从图 12-4 可以看到，当前目录面板除了包括当前目录工具条的显示框和两个工具按钮外，还提供了更多工具按钮，它们在图 12-4 中紧邻着切换到父目录的工具按钮，从左向右依次是：新建目录按钮，文件查找按钮，可视化目录显示按钮，报告按钮（点击下拉三角形会显示多种报告按钮）。



新建目录按钮用于在当前目录下新建一个目录。

报告按钮包括多种 MATLAB 报告按钮，如 M-Lint 报告按钮等。读者可以自行参考 MATLAB 帮助，了解这些报告按钮的功能，本书不作介绍。

文件查找按钮可以在指定的目录下查询符合指定条件的 MATLAB 文件。单击文件查找按钮，弹出如图 12-5 所示的文件查找窗口。

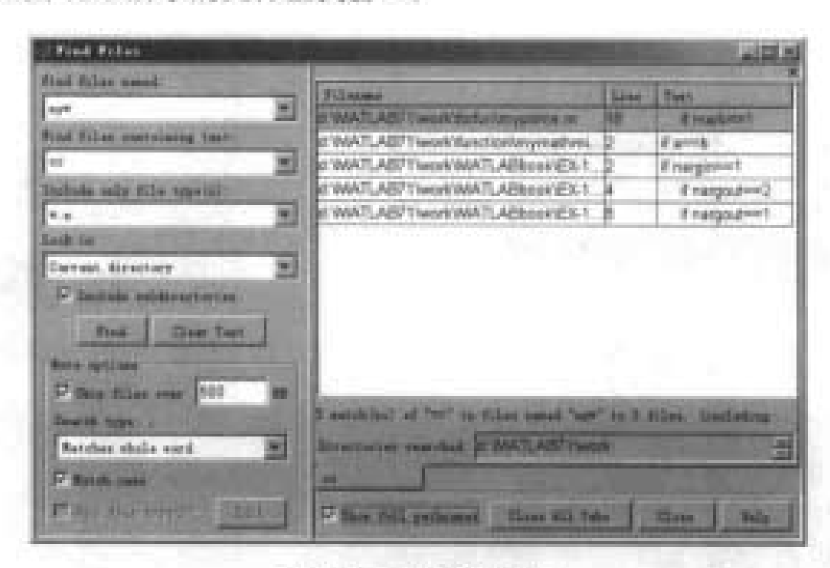


图 12-5　文件查找窗口

文件查找窗口提供了多项可定制的查找选项：文件名（支持通配符）、文件包含文字、文件类型、查找目录等，除此之外，MATLAB 文件查找中还可以忽略大文件（如图 12-5 所示的查找中忽略大于 500MB 的文件）、匹配查找字符串的大小写、忽略特定的文件类型（需要设置查找文件类型为 All Files(*)）。

例如图 12-5 显示就是在当前目录（d:\MATLAB71\work）及其所有子目录中查找文件名以 my 开头（my*表示以 my 开头，后面可以接任意数目的任意字符），文件中包括“==”的 m 类型的文件。查找结果显示了所有符合查找条件的文件中包含查找字符串的行的详细信息。

MATLAB 的文件查找窗口会记录最近 10 次的查找历史，显示在文件查找窗口右侧下方的标签中，这样用户可以通过切换标签快速浏览查找历史记录。

## 12.1.3　可视化目录显示

当前目录面板中文件查找按钮右侧就是可视化目录显示按钮。单击可视化目录显示按钮，出现如图 12-6 所示的目录可视化显示窗口。

目录可视化显示窗口的第一行显示当前目录的子目录及其包含的文件数，并可快捷链接到这些子目录和当前目录的父目录。接着提供了一些目录可视化显示窗口显示内容的开关选项：按 Contents.m 排序、显示动作、显示结果缩略图、显示文件大小、显示文件属于函数还是脚本。默认情况下，这些都是开关选项都是开启的。因此下方紧邻的显示结果中



包含了这些信息。这些显示结果中蓝色的文字部分都是快捷链接，单击可以链接打开或运行、删除相应的文件。

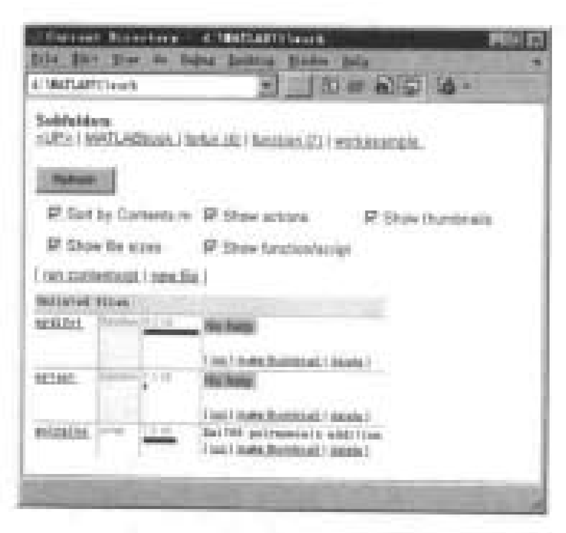


图 12-6　目录可视化显示窗口

## 12.1.4　当前目录设置

对于当前目录工具条和当前目录面板的显示内容，MATLAB 可以自定义设置。

在当前目录面板下，选择 File 菜单的 Preferences…项，可以打开当前目录参数设置窗口，如图 12-7 所示。用户也可以在 MATLAB 主界面下选择 File 菜单的 Preferences…项，打开参数设置窗口后在窗口左侧选择 Current Directory 项，这样也可以打开当前目录参数设置窗口。

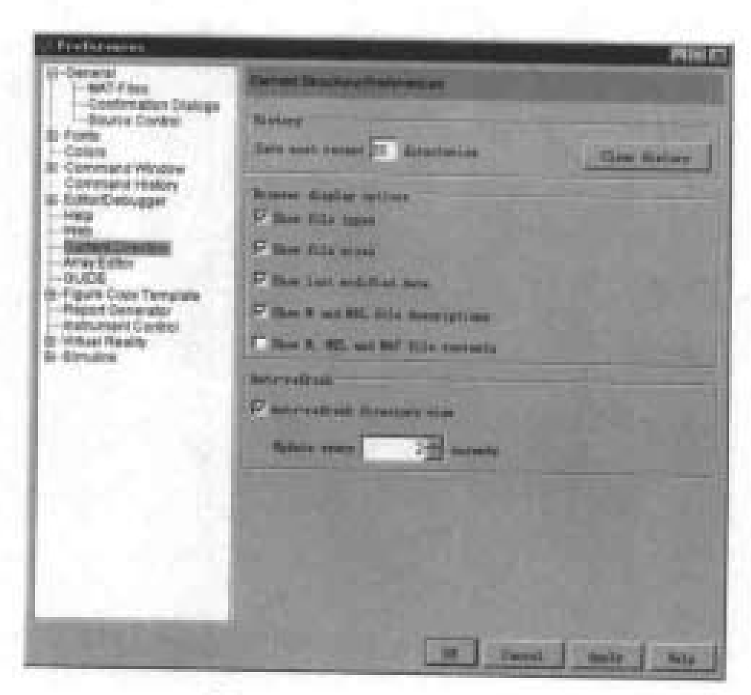


图 12-7　当前目录设置窗口

从图 12-7 可见，通过当前目录设置窗口，用户可以设置当前目录工具条保存的目录历史记录项数目，当前目录面板显示项，以及当前目录自动刷新间隔。

### 12.1.5 命令窗口目录操作命令

MATLAB 提供了许多可以在命令窗口执行的目录操作命令，如表 12-1 所示。

表 12-1 目录操作命令

| 命令 | 说明 |
|---|---|
| pwd | 返回当前目录 |
| matlabroot | 返回 MATLAB 安装目录 |
| dir | 显示当前目录的子目录和文件 |
| isdir(var) | 判断某个变量是否为目录 |
| cd yourdir | 把当前目录变换到 yourdir 指定的目录 |
| what | 显示当前目录下的 MATLAB 文件 |
| which yourfile | 返回 yourfile 指定的文件的全路径 |
| mkdir newdir | 创建 newdir 指定的新目录 |
| rmdir yourdir | 删除 yourdir 指定的目录 |

这些操作命令的用法，如例 12-1 所示。

**例 12-1** 目录操作命令。

**解**：在命令窗口输入：

```
>> a=pwd  %返回当前目录
a =
d:\MATLAB71\work
>> matlabroot     %返回MATLAB安装目录
ans =
D:\MATLAB71
>> isdir(a)
ans =
     1
>> what

M-files in the current directory d:\MATLAB71\work

mydlfct     mytest     polyplus

MAT-files in the current directory d:\MATLAB71\work

data1      data2      data3

>> which myprime
D:\MATLAB71\work\forfun\myprime.m
>> mkdir testdir
>> dir

.            MATLABbook    data2.mat    forfun        html          mygui.rar
polyplus.m   workexample
```



```
    ..           data1.mat    data3.mat    function     mydlfct.m    mytest.m
testdir

    >> rmdir testdir
    >> dir

    .            MATLABbook   data2.mat    forfun       html         mygui.rar
polyplus.m
    ..           data1.mat    data3.mat    function     mydlfct.m    mytest.m
workexample

    >> cd ..         %切换当前目录到其父目录
    >> pwd
    ans =
    d:\MATLAB71
    >> cd(a)
    >> pwd
    ans =
    d:\MATLAB71\work
```

## 12.2 MATLAB 搜索路径

### 12.2.1 MATLAB 文件运行搜索过程

MATLAB 运行的文件必须位于当前目录或 MATLAB 搜索路径下。MATLAB 搜索路径实际上由一系列目录组成，这些目录下面包括了所有 MATLAB 命令窗口可以运行的文件。

用户在运行自己编写的 MATLAB 文件时，除了把当前目录切换到这些文件所在的目录外，也可以通过把这些目录添加到 MATLAB 搜索路径，这样也可以在命令窗口下成功调用这些文件，而且，设置搜索路径的方法还是一劳永逸的。

当用户在命令窗口键入一个命令（例如 foo）时：

（1）MATLAB 先在当前工作区变量空间中查找是否存在变量 foo，如果存在，则返回变量 foo 的结果；

（2）否则，MATLAB 将在当前目录下查找是否存在 foo.m 文件；如果存在，则运行 foo.m 文件；

（3）否则，MATLAB 将按顺序搜索 MATLAB 搜索路径的各个目录，直到查找到 foo.m 文件则运行它。

有些情况下，在 MATLAB 搜索路径的所有目录下可能存在多个名为 foo.m 的文件，这时候 MATLAB 只执行查找到的第一个 foo.m。用户如果希望执行特定的 foo.m，可以通过切换当前目录，或者通过 run 命令和 foo.m 的全路径按照 run direcotry/foo.m 这种语法格式来运行 foo.m。

MATLAB 搜索路径信息存储在$matlabroot/toolbox/local 目录下的 pathdef.m 文件中。MATLAB 启动时，首先在当前文件夹查找是否存在 pathdef.m，如果存在则装载此路径设置，否则调用$matlabroot/toolbox/local 目录下 pathdef.m 来设置 MATLAB 工作时的搜索路径。




### 12.2.2 搜索路径设置

查看和设置 MATLAB 搜索路径，都可以通过选择 File 菜单的 Set Path…项打开路径设置窗口来进行，如图 12-8 所示。

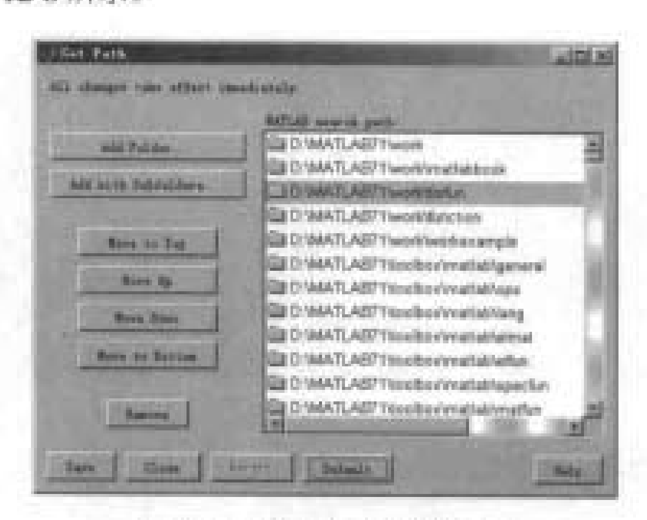


图 12-8 MATLAB 路径设置窗口

通过 MATLAB 路径设置窗口对 MATLAB 搜索路径进行的任何改变，都会立即生效。设置窗口中提供了许多设置按钮：添加一个目录、添加一个目录及其所有子目录，将选定目录移动到顶端、向上移动、向下移动、移动到底部，将选定目录删除出搜索路径，保存修改、关闭设置窗口，取消上一次修改，恢复为 MATLAB 默认的搜索路径，帮助等。

大部分情况下，用户只要选择添加目录及其子目录的按钮，选择要添加的文件夹，然后路径显示中就会出现用户指定的目录及其子目录，如果需要设置顺序的话，选择这些目录，移动其位置即可。最后，保存设置并关闭设置窗口。

### 12.2.3 搜索路径设置命令

除了通过设置窗口设置 MATLAB 搜索路径外，MATLAB 也提供了许多路径设置命令，如表 12-2 所示。由于通过设置窗口设置搜索路径的方法更具有交互性和易用性，因此这些命令使用不是很多，有兴趣的读者可以参考 MATLAB 帮助文件学习。

表 12-2 MATLAB 搜索路径设置命令

| 命 令 | 说 明 |
|---|---|
| path | 显示和改变 MATLAB 搜索路径 |
| pathtool | 打开 MATLAB 搜索路径设置窗口 |
| restoredefaultpath | 将 MATLAB 搜索路径设置为默认值 |
| rmpath | 从 MATLAB 搜索路径中删除指定的目录 |
| addpath | 向 MATLAB 搜索路径中添加指定的目录 |
| savepath | 将当前 MATLAB 搜索路径保存到 pathdef.m 文件中 |



## 12.3 文件管理

MATLAB 中最基本最常用的文件操作是数据的导入和导出操作。数据导入是指把本地磁盘文件或剪贴板中的数据装载到 MATLAB 工作空间的过程，数据导出则是指把 MATLAB 工作空间的变量保存到本地磁盘文件的过程。

MATLAB 的数据导入和导出操作支持多种数据格式，它们包括：文本数据、图形数据、音频和视频数据、电子表格数据和科学数据。针对各种不同类型的数据格式，MATLAB 提供了相应的导入和导出命令，用于处理各种相应的数据文件，如表 12-3 到表 12-5 所示。

### 12.3.1 文本数据

文本文件中的数据是按照 ASCII 码存储的字符或数字，它们可以显示在任何文本编辑器中，MATLAB 中文本数据的导入和导出函数如表 12-3 所示。

表 12-3 文本数据的导入和导出函数

| 函 数 | 说 明 |
|---|---|
| csvread | 以逗号为分隔符，将文本数据读入 MATLAB 工作区 |
| csvwrite | 以逗号为分隔符，将 MATLAB 工作区变量写入文本文件 |
| dlmread | 以指定的 ASCII 字符为分隔符，将文本数据读入 MATLAB 工作区 |
| dlmwrite | 以指定的 ASCII 字符为分隔符，将 MATLAB 工作区变量写入 ASCII 文本文件 |
| textread | 按照指定格式从文本文件读入数据到 MATLAB 工作区 |
| textwrite | 按照指定格式将 MATLAB 工作区变量写入文本文件 |

### 12.3.2 图形、音频和视频数据

MATLAB 中支持多种格式的图形文件，包括 TIFF、GIF、JPEG、PNG 等多种标准格式。关于 MATLAB 中图形文件的导入导出，详见本书第 26，27 章，这里先不作讨论。

MATLAB 中对音频和视频文件也提供了很好的支持，可以处理 NeXT/SUN 声音文件、Microsoft WAVE 声音文件、AVI 文件，这部分内容详见本书第 26 章。

### 12.3.3 电子表格数据

最常见的电子表格文件是 Microsoft Excel 生成的.xls 文件和 Lotus123 生成的.wk1 文件，它们在许多统计分析领域应用广泛，MATLAB 处理这些电子表格文件的函数如表 12-4 所示。

表 12-4 电子表格数据的导入和导出函数

| 电子表格生成程序 | 函 数 | 说 明 |
|---|---|---|
| Microsoft Excel | xlsinfo | 检查文件是否包含.xls 电子表格 |
| | xlsread | 从.xls 文件读入数据到 MATLAB 工作区 |
| | xlswrite | 把 MATLAB 工作区变量写回到.xls 文件中 |



续表

| 电子表格生成程序 | 函　数 | 说　明 |
|---|---|---|
| Lotus123 | wk1read | 从.wk1 文件读入数据到 MATLAB 工作区 |
| | wk1write | 把 MATLAB 工作区变量写回到.wk1 文件中 |

### 12.3.4　科学标准格式数据

MATLAB 支持多种科学标准文件格式，包括 CDF（Common Data Format），FITS（Flexible Image Transport System），HDF（Hierarchical Data Format），Band-Interleaved Data 等格式。这些格式的数据导入和导出函数如表 12-5 所示。

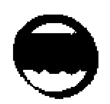
表 12-5　科学标准数据的导入和导出函数

| 文件格式 | 函　数 | 说　明 |
|---|---|---|
| CDF | cdfepoch | 从字符串或数值数据创建 cdfepoch 对象 |
| | cdfinfo | 返回 CDF 文件信息 |
| | cdfread | 读入 CDF 文件 |
| | cdfwrite | 写回 CDF 文件 |
| | todatenum | 把 cdfepoch 对象转换成 MATLAB 日期数值 |
| FITS | fitsinfo | 返回 FITS 文件信息 |
| | fitsread | 读入 FITS 文件 |
| HDF | hdfinfo | 返回 HDF4 或 HDF-EOS 文件信息 |
| | hdfread | 读入 HDF4 文件 |
| | hdftool | 打开 HDF4 数据导入向导 |
| | hdf5info | 返回 HDF5 文件信息 |
| | hdf5read | 读入 HDF5 文件 |
| | hdf5write | 写回 HDF5 文件 |
| Band-Interleaved Data | multibandread | 从文件读入 Band-Interleaved 数据 |
| | multibandwrite | 把 Band-Interleaved 数据写回文件 |

### 12.3.5　数据导入向导

除了通过 MATLAB 提供的数据导入和导出命令处理数据文件外，MATLAB 还提供了数据导入向导，可以交互式地导入用户数据。用户通过选择 File 菜单的 Import Data…项就可以打开数据导入向导。向导通过交互式的方式提示用户选择特定文件类型的数据文件，并指定许多导入设置项。数据导入向导，对于一般的 MATLAB 使用者是很方便易用的。

对于许多专业领域处理各种相应格式数据的编程用户，数据导入和导出函数无疑是功能强大好用的。但对于一般的 MATLAB 使用者，遇到大规模本地文件数据或专业格式数据的情况比较少，即使遇到，一般也可以用数据导入向导简单便捷地完成数据文件操作。



## 12.3.6 因特网文件处理

MATLAB 除了可以处理本地数据文件外，也能和因特网进行交互，处理因特网上的文件。MATLAB 中和因特网交互相关的命令如表 12-6 所示。

表 12-6 因特网交互命令

| 功　能 | 命　令 | 说　明 |
|---|---|---|
| 文件压缩、提取 | zip | 将文件压缩为 zip 格式 |
| | unzip | 提取 zip 文件内容 |
| | gzip | 将文件压缩为 gzip 格式 |
| | gunzip | 解压缩 gzip 格式的文件中的内容 |
| | tar | 将文件压缩为 tar 格式 |
| | untar | 提取 tar 文件内容 |
| URL | urlread | 从 URL 读取内容 |
| | urlwrite | 将 URL 的内容写回文件中 |
| E-mail | sendmail | 发送 E-mail |
| FTP | ftp | 连接到 ftp 服务器，创建一个 ftp 对象 |
| | dir | 列表 ftp 服务器目录下的内容 |
| | cd | 改变 ftp 服务器目录 |
| | rename | 对 ftp 服务器上的文件、目录重命名 |
| | delete | 删除 ftp 服务器上的文件 |
| | mkdir | 在 ftp 服务器上创建目录 |
| | rmdir | 删除 ftp 服务器上的目录 |
| | ascii | 设置 ftp 传送方式为 ascii |
| | binary | 设置 ftp 传送方式为 binary |
| | mget | 从 ftp 服务器下载文件 |
| | mput | 向 ftp 服务器上传文件 |
| | close | 关闭到 ftp 服务器的连接 |

从表 12-6 可以看出，通过 MATLAB 提供的网络交互命令，用户可以通过 MATLAB 访问因特网链接、发送电子邮件和 FTP 服务器对话等。

## 12.3.7 低级文件 I/O

另外，MATLAB 中还有大量低级文件 I/O 指令，如表 12-7 所示，这些指令可以对多种类型的数据文件进行操作。低级文件 I/O 指令是基于 ANSI 标准 C 库的，它们和 C 语言中的相应指令用法一样，C 编程的用户看到这些指令会很熟悉。

低级文件 I/O 指令，能够满足大部分本地文件和 MATLAB 工作区的交互需要，可以解决大部分文件数据导入导出的问题，各种层次的 MATLAB 编程用户都应该学习掌握。





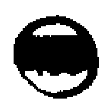
表 12-7　低级文件 I/O 指令

| 命　　令 | 说　　明 |
|---|---|
| fopen | 打开文件或获取已打开文件的信息 |
| fclose | 关闭文件 |
| feof | 测试光标是否到达文件末尾 |
| ferror | 查询文件操作错误 |
| ftell | 返回文件中光标位置 |
| fseek | 设置文件中光标位置 |
| frewind | 将文件中光标位置移动到文件头 |
| fscanf | 按指定格式读入文件中数据 |
| fprintf | 按指定格式将数据写回文件 |
| fread | 以二进制方式读入文件中的数据 |
| fwrite | 以二进制方式将数据写回文件 |
| fgetl | 返回不包括行尾终止符的字符串 |
| fgets | 返回包括行尾终止符的字符串 |

本节只是简要概述了 MATLAB 中对各种数据文件的导入导出指令。这些相关函数参数复杂，功能丰富，不同专业领域的编程用户的需求相差很大，因此建议读者就自己感兴趣的部分，仔细参考 MATLAB 联机帮助。

## 12.4　小结

本章讲解了 MATLAB 中目录、路径管理和文件 I/O。其中目录、路径管理的内容解决了 MATLAB 中文件运行的部分问题，其中通过图形窗口交互设置的方法，对于所有 MATLAB 用户来说都是应该熟练掌握的。相应的设置命令对编程用户提供了很多方便，用户可以有针对性地了解学习。文件 I/O 部分概述了各种数据导入导出的 MATLAB 指令，这部分内容具有较大的专业特异性，因此用户可以根据自己的实际使用情况选择学习，但其中低级文件 I/O 指令用途广泛，且能解决大部分常见的数据导入导出问题，建议用户了解和基本掌握。



# 第 13 章 MATLAB 中的时间

本章介绍 MATLAB 中与系统时间、日期相关的知识。通过本章学习，读者可以了解到 MATLAB 中时间日期的三种表示格式及其相互转换，获取当前日期时间的函数的使用方法，以及在程序中定时从而确定程序实际运行时间的方法。

## 13.1 日期和时间

### 13.1.1 日期时间的三种表示格式

MATLAB 中表示日期时间有三种格式：日期字符串、连续的日期数值和日期向量，如表 13-1 所示。

表 13-1 MATLAB 中日期时间的三种格式

| 日期时间格式 | 举 例 |
| --- | --- |
| 日期字符串 | 13-Jan-2006 |
| 连续的日期数值 | 732691 |
| 日期向量 | 2006 1 13 20 57 11.515 |

日期字符串格式是 MATLAB 命令行下最常用的，它有多种输出样式：例如 13-Jan-2006 20:59:03、13-Jan-2006、01/13/06 等，这些在 13.1.4 节中会详细介绍。

MATLAB 中，连续的日期数值格式是以公元元年 1 月 1 日为起点的，用一个数值表示当前时间到这个起点的时间距离。

例如 2006 年 1 月 13 日 21 点，用日期字符串表示是'13-Jan-2006 21:00'，而用连续的日期数值格式表示则是 732690.875。其中整数部分 732690 代表这一天距离公元元年 1 月 1 日

已经过去了 731885 天，小数部分 0.875 代表这一天已经过去了 7/8，即 21 个小时。

MATLAB 内部对日期时间进行存储和计算时，就是用这种连续的日期数值格式表示的，因此本质上，日期时间也是一种数值类型。在对多个日期时间进行计算时，采用这种格式可以减少运算中的转换步骤，提高运算效率。

日期向量格式是用一个包括 6 个数字的数组来表示日期时间，其元素顺序依次为[year month day hour minute second]，它是某些 MATLAB 内部函数的返回和参数输入格式，一般不用于日期时间的运算中。

### 13.1.2 获取当前日期时间的函数

MATLAB 中获取当前日期时间的函数有：

（1）date 函数按照日期字符串格式返回当前的系统日期；

（2）now 函数按照连续的日期数值格式返回当前的系统时间；

（3）clock 函数按照日期向量格式返回当前的系统时间。

**例 13-1** 获取系统当前日期和时间。

**解：**在命令窗口输入：

```
>> date
ans =
13-Jan-2006
>> now
ans =
  7.3269e+005
>> clock
ans =
       2006           1          13          21          23      37.812
```

MATLAB 中还有分别提取年、月、日、时、分、秒信息的函数。分别是 year，month，day，hour，minute，second，它们都以日期字符串格式或者连续的日期数值格式表示的时间为参数，返回该时间的年、月、日、时、分、秒信息。

**例 13-2** 日期时间局部信息提取函数。

**解：**在命令窗口输入：

```
>> str=date
str =
14-Jan-2006
>> year(str)
ans =
        2006
>> month(str)
ans =
     1
>> day(str)
ans =
    14
>> minute(str)
ans =
     0
```



### 13.1.3　日期格式转换

MATLAB 提供了三个函数用于在三种日期时间表示格式之间进行转换。

（1）datestr 函数：把某种日期时间格式转换成日期字符串格式输出。

（2）datenum 函数：把某种日期时间格式转换成连续的日期数值输出。

（3）datevec 函数：把某种日期时间格式转换成时间向量格式输出。

**例 13-3**　MATLAB 中的日期格式转换函数。

**解**：在命令窗口输入：

```
>> num=now
num =
  7.3269e+005
>> datestr(now)
ans =
13-Jan-2006 21:38:53
>> vec=datevec(num)
vec =
       2006          1         13         21         38     47.843
>> datenum(vec)
ans =
  7.3269e+005
```

datenum 函数能够转换生成连续的日期数值格式，这在日期计算中非常重要。datenum 函数可以采用多种样式的日期字符串格式作为输入参数，最常用的输入样式有：'dd-mmm-yyyy'，'mm/dd/yyyy'，'dd-mmm-yyyy, hh:mm:ss.ss'。其中年份数字可以采用'yy'形式的两位输入则代表 20yy 年，若省略年份则默认为当前年。

需要注意的是，在'mm/dd/yyyy'的纯数字样式输入中，第一部分代表月，第二部分代表日，这和'dd-mmm-yyyy'中的顺序是相反的。

### 13.1.4　datestr 转换函数输出样式控制

实际应用中 datestr 函数可以指定输出字符串的样式。

datestr(*D*,*F*)把日期字符串 ***D***、连续的日期数值 ***D***、或日期向量 ***D*** 转换成由 ***F*** 指定样式的日期字符串格式。***F*** 可以是一个数字，也可以是一个字符串。

表 13-2 列出了对 ***F*** 所有允许的取值或字符串，并举例加以说明。

表 13-2　日期字符串的各种输出样式

| *F*（数字） | *F*（字符串） | 举　例 |
|---|---|---|
| 0 | 'dd-mmm-yyyy HH:MM:SS' | 14-Jan-2006 09:50:19 |
| 1 | 'dd-mmm-yyyy' | 14-Jan-2006 |
| 2 | 'mm/dd/yy' | 01/14/06 |
| 3 | 'mmm' | Jan |
| 4 | 'm' | J |
| 5 | 'mm' | 01 |

续表

| F（数字） | F（字符串） | 举　例 |
|---|---|---|
| 6 | 'mm/dd' | 01/14 |
| 7 | 'dd' | 14 |
| 8 | 'ddd' | Sat |
| 9 | 'd' | S |
| 10 | 'yyyy' | 2006 |
| 11 | 'yy' | 06 |
| 12 | 'mmmyy' | Jan06 |
| 13 | 'HH:MM:SS' | 09:50:19 |
| 14 | 'HH:MM:SS PM' | 9:50:19 AM |
| 15 | 'HH:MM' | 09:50 |
| 16 | 'HH:MM PM' | 9:50 AM |
| 17 | 'QQ-YY' | Q1-06 |
| 18 | 'QQ' | Q1 |
| 19 | 'dd/mm' | 14/01 |
| 20 | 'dd/mm/yy' | 14/01/06 |
| 21 | 'mmm.dd,yyyy HH:MM:SS' | Jan.14,2006 09:50:19 |
| 22 | 'mmm.dd,yyyy' | Jan.14,2006 |
| 23 | 'mm/dd/yyyy' | 01/14/2006 |
| 24 | 'dd/mm/yyyy' | 14/01/2006 |
| 25 | 'yy/mm/dd' | 06/01/14 |
| 26 | 'yyyy/mm/dd' | 2006/01/14 |
| 27 | 'QQ-YYYY' | Q1-2006 |
| 28 | 'mmmyyyy' | Jan2006 |
| 29 | 'yyyy-mm-dd' | 2006-01-14 |
| 30 | 'yyyymmddTHHMMSS' | 20060114T095019 |
| 31 | 'yyyy-mm-dd HH:MM:SS' | 2006-01-14 09:50:19 |

## 13.2　程序中应用的计时函数

在编写 MATLAB 代码的时候，经常需要获知代码执行的实际时间，这就需要在程序中用到计时函数。MATLAB 提供了 cputime，tic/toc，etime 三种方法。

（1）cputime 方法

返回 MATLAB 启动以来的 CPU 时间，可以在程序代码执行前保存当时的 CPU 时间，然后在程序代码执行结束后用 cputime 减去之前保存的数值，就可以获取程序实际运行的时间。

（2）tic/toc 方法

tic 方法与 toc 方法类似，tic 用在程序代码首部，启动一个计时器；toc 放在程序代码





末尾，终止计时器并返回 tic 启动以来的总时间。

（3）etime 方法

etime(*t*1,*t*2)用来计算两个日期向量 *t*1 和 *t*2 之间的时间差，结合前面讲到的 clock 函数也可以用来确定程序代码运行时间。

例 13-4　程序中的定时函数。

**解**：在命令窗口输入：

```
>> t=cputime;peaks(40);cputime-t

ans =
    0.1563
>> tic;peaks(40);toc

Elapsed time is 0.118182 seconds.
>> t=clock;peaks(40);etime(clock,t)

ans =
    0.0630
```

例 13-4 中用这三种方法分别计算了执行 peaks(40)（MATLAB 内置的一个二维绘图函数）消耗的时间。

## 13.3　小结

本章讲述了 MATLAB 中日期时间的三种表示格式及其相互转化，尤其对日期字符串格式的多种输出样式进行了重点讲解，另外，对三种格式各自的应用情况也作了简单的介绍，最后，对程序编写中经常用到的计时函数进行了分类介绍和举例。

# 第 14 章 矩阵代数

在很多工程应用领域，都会遇到矩阵分析、线性方程组的求解等问题，这些属于线性代数学科的研究范围，本章讲解用 MATLAB 解决这些线性代数基本问题的方法。

## 14.1 矩阵分析

矩阵是线性代数研究的基本元素，实际上相当于 MATLAB 中的普通二维数组。矩阵分析主要是研究矩阵的各种特性及其表征方法。

### 14.1.1 矩阵的行列式

矩阵的行列式是一个数值，它可以用来表示矩阵是否奇异（矩阵行列式等于 0），这主要用在线性方程组特性分析上。MATLAB 中求解矩阵行列式的函数是 det。

例 14-1 矩阵行列式。

**解**：在命令窗口输入：

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> det(A)
ans =
  -360
```



## 14.1.2　矩阵的逆

矩阵的逆是线性代数中的一个重要概念。如果矩阵 ***A*** 的逆矩阵为 ***B***，那么 ***B*A=A*B=I***，即一个矩阵和它的逆矩阵左乘或者右乘结果都是单位矩阵。MATLAB 中可以通过函数 inv 求解矩阵的逆。从 ***B*A=A*B=I*** 容易看到，只有方阵才具有逆矩阵。

矩阵的逆在求解线性方程组时是重要的，对于一般的给定线性方程组 ***A*X=b***，其解就可以通过 ***X***=inv(***A***)****b*** 求得。

需要注意的是，对于严重病态的矩阵或奇异矩阵，inv 求解时会出警告提示，因为这时候其逆矩阵本来就不存在，或者非常容易受扰动而使得求解不精确。

对于一般的长方形矩阵 ***A***，***A*X=I*** 和 ***X*A=I*** 中至少有一个没有解，因此长方形矩阵 ***A*** 没有逆矩阵。但 MATLAB 中可以用 pinv 函数求解长方形矩阵 ***A*** 的伪逆矩阵 ***B***=pinv(***A***)，***B*** 一定满足 ***B* A= I*** 或 ***A*B=I*** 中的一个等式。

**例 14-2**　矩阵的逆。

**解**：在命令窗口输入：

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> B=inv(A)
B =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
>> A*B
ans =
    1.0000         0   -0.0000
   -0.0000    1.0000         0
    0.0000         0    1.0000
>> B*A
ans =
    1.0000         0   -0.0000
         0    1.0000         0
         0    0.0000    1.0000
>> C=rand(2,4)
C =
    0.8214    0.6154    0.9218    0.1763
    0.4447    0.7919    0.7382    0.4057
>> D=pinv(C)
D =
    1.3909   -1.1648
   -0.8303    1.3844
    0.5848   -0.1305
   -0.9680    1.2767
>> C*D
ans =
    1.0000   -0.0000
         0    1.0000
>> D*C
```

```
ans =
    0.6246   -0.0664    0.4224   -0.2274
   -0.0664    0.5854    0.2566    0.4153
    0.4224    0.2566    0.4428    0.0501
   -0.2274    0.4153    0.0501    0.3473
```

### 14.1.3 矩阵的秩

矩阵的秩反映了矩阵的各行向量之间和各列向量之间的线性依赖关系。对于一个满秩方阵（即秩等于行数或列数），其各行向量和各列向量都线性无关。MATLAB 中求解矩阵秩的函数是 rank。

**例 14-3** 矩阵的秩。

**解：** 在命令窗口输入：

```
>> rank(magic(3))
ans =
     3
>> rank(eye(5))
ans =
     5
>> rank(zeros(2,4))
ans =
     0
```

### 14.1.4 矩阵的范数和条件数

矩阵条件数是用来刻画矩阵病态程度的关键表征量。矩阵的条件数越大，代表矩阵病态程度越严重。线性方程组 $A*X=b$ 中，如果系数矩阵 $A$ 严重病态，其精确求解将是很困难的。

MATLAB 中求解矩阵范数的函数是 norm，norm(*A*,*p*)求解矩阵 $A$ 的 $p$-范数。

（1）1-范数 norm(*A*,1)实际上返回矩阵 $A$ 列向元素和的最大值 max(sum(abs(*A*)))；

（2）2-范数 norm(*A*,2)返回矩阵 $A$ 的最大奇异值 max(svd(*A*))；

（3）无穷范数 norm(*A*,inf)返回矩阵 $A$ 行向元素和的最大值 max(sum(abs(*A*')))。

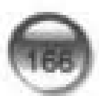
norm(*A*)相当于 norm(*A*,2)，求解矩阵 $A$ 的 2-范数。

矩阵的条件数是在矩阵的逆和矩阵范数的基础上定义的，MATLAB 中求解矩阵条件数的函数是 cond。cond(*A*,*p*)等于 norm(*A*,*p*)*norm(inv(*A*),*p*)。

**例 14-4** 矩阵的范数和条件数。

**解：** 在命令窗口输入：

```
>> norm(magic(3))
ans =
   15.0000
>> norm(inv(magic(3)))
ans =
    0.2887
>> cond(magic(3))
```





```
ans =
    4.3301
>> cond(inv(magic(3)))
ans =
    4.3301
>> cond(zeros(3,3))
ans =
   Inf
```

从例 14-4 及前面文字讲解内容可以知道，一个矩阵和它的逆矩阵一定具有相同的条件数。另外，对于秩为 0 或非常接近 0 的奇异矩阵，其条件数会非常大，亦即矩阵病态程度很严重。

### 14.1.5　矩阵的特征值、特征向量和特征多项式

对于 $n*n$ 的矩阵 $A$，如果存在 $n$ 维列向量 $x$ 和标量数值$\lambda$使得 $A*x=\lambda*x$，那么，把 $x$ 称为矩阵 $A$ 的特征向量，$\lambda$称为矩阵 $A$ 的特征值。MATLAB 中可以通过 eig 函数求解矩阵的特征值和特征向量：

（1）***d*=eig(*A*)**返回 $n*n$ 矩阵 $A$ 的 $n$ 个特征值；

（2）**[*V*,*D*]=eig(*A*)**返回以矩阵 $A$ 的特征向量为列的矩阵 $V$ 和以矩阵 $A$ 的特征值为对角元素的矩阵 $D$。

**poly(*A*)**生成矩阵 $A$ 的特征多项式，即 $\det(\lambda*I-A)$这一$\lambda$的 $n$ 阶多项式。矩阵 $A$ 的特征多项式的根，实际上就是矩阵 $A$ 的特征值。

**例 14-5**　矩阵的特征值、特征向量和特征多项式。

**解：**在命令窗口输入：

```
>> A=rand(3,3)
A =
    0.9355    0.8936    0.8132
    0.9169    0.0579    0.0099
    0.4103    0.3529    0.1389
>> d=eig(A)
d =
    1.7342
   -0.4063
   -0.1957
>> [V,D]=eig(A)
V =
   -0.8319   -0.4372    0.2126
   -0.4569    0.8686   -0.7911
   -0.3150   -0.2332    0.5736
D =
    1.7342         0         0
         0   -0.4063         0
         0         0   -0.1957
>> A*V
ans =
   -1.4426    0.1776   -0.0416
   -0.7923   -0.3529    0.1548
```

```
   -0.5463    0.0947   -0.1122
>> V*D
ans =
   -1.4426    0.1776   -0.0416
   -0.7923   -0.3529    0.1548
   -0.5463    0.0947   -0.1122
>> roots(poly(A))
ans =
    1.7342
   -0.4063
   -0.1957
```

### 14.1.6 矩阵的标准正交基

一个矩阵通过其每一列向量的线性运算，可以派生出一个向量空间，这称之为矩阵的线性空间。每一个矩阵的线性空间下所有的向量，实际上只需要通过一组基向量的线性运算就可以产生。这样的最少个数的一组基向量称为该空间的基，如果这些向量正好长度为1，彼此正交，则称为标准正交基。

MATLAB 中可以通过 orth 函数产生矩阵 ***A*** 的线性空间的一组标准正交基，即若 ***B***=orth(***A***)，则 ***B*** 的列向量组成了矩阵 ***A*** 的线性空间的一组标准正交基，于是 ***B'*B***=eye(rank(***A***))。

**例 14-6** 矩阵的标准正交基。

**解**：在命令窗口输入：

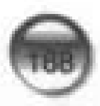
```
>> A=rand(3)
A =
    0.4660    0.5252    0.8381
    0.4186    0.2026    0.0196
    0.8462    0.6721    0.6813
>> B=orth(A)
B =
   -0.6265    0.6290   -0.4603
   -0.2164   -0.7077   -0.6726
   -0.7488   -0.3218    0.5795
>> B'*B
ans =
    1.0000         0   -0.0000
         0    1.0000    0.0000
   -0.0000    0.0000    1.0000
```

### 14.1.7 矩阵分解

处理各种矩阵相关问题时，经常会利用矩阵分解的方法。MATLAB 中对各种经典的矩阵分解都提供了相关的函数。

1．LU 分解

LU 分解，是把矩阵 ***A*** 分解为两个矩阵的乘积，其中一个是下三角矩阵置换后的矩阵，





另一个是上三角矩阵。MATLAB中通过函数lu可以实现矩阵的LU分解：

（1）[*L*,*U*]=lu(*A*)把矩阵 *A* 分解为下三角矩阵的置换矩阵 *L* 和上三角矩阵 *U*，满足 *A*=*L***U*；

（2）[*L*,*U*,*P*]=lu(*A*)的分解结果中 *L* 是一个下三角矩阵，*U* 是一个上三角矩阵，*P* 是一个置换矩阵，满足 *L***U* = *P***A*。

**例 14-7** LU分解。

**解**：在命令窗口输入：

```
>> A=rand(3)
A =
    0.3795    0.7095    0.1897
    0.8318    0.4289    0.1934
    0.5028    0.3046    0.6822
>> [L,U]=lu(A)
L =
    0.4562    1.0000         0
    1.0000         0         0
    0.6045    0.0883    1.0000
U =
    0.8318    0.4289    0.1934
         0    0.5138    0.1014
         0         0    0.5563
>> L*U
ans =
    0.3795    0.7095    0.1897
    0.8318    0.4289    0.1934
    0.5028    0.3046    0.6822
>> [L,U,P]=lu(A)
L =
    1.0000         0         0
    0.4562    1.0000         0
    0.6045    0.0883    1.0000
U =
    0.8318    0.4289    0.1934
         0    0.5138    0.1014
         0         0    0.5563
P =
     0     1     0
     1     0     0
     0     0     1
```

### 2. Cholesky分解

一个对称正定矩阵，可以分解为一个下三角矩阵和一个上三角矩阵的乘积，这种分解称为Cholesky分解。MATLAB中通过chol函数实现矩阵的Cholesky分解。

*R*=chol(*A*)得到一个上三角矩阵，满足 *R*'**R*=*A*。

因为chol只能分解对称正定矩阵，因此使用chol之前最好通过eig命令检查矩阵 *A* 的所有特征值是否为正（即矩阵是否正定）。

**例 14-8** Cholesky分解。

**解**：在命令窗口输入：

```
>> A=[1 2 2;2 6 8;2 8 13]
A =
     1     2     2
     2     6     8
     2     8    13
>> eig(A)
ans =
    0.0859
    1.2477
   18.6664
>> R=chol(A)
R =
    1.0000    2.0000    2.0000
         0    1.4142    2.8284
         0         0    1.0000
>> R'*R
ans =
     1     2     2
     2     6     8
     2     8    13
```

### 3. QR 分解

QR 分解是把矩阵分解为一个正交矩阵和一个上三角矩阵的乘积。MATALB 中实现 QR 分解的命令是 qr。

[***Q***,***R***]=qr(***A***)把矩阵 ***A*** 分解为正交矩阵 ***Q*** 和上三角矩阵 ***R***，满足 ***A***=***Q********R***。

QR 分解不仅适用于分解方阵，也可以分解长方形矩阵。

**例 14-9** QR 分解。

**解**：在命令窗口输入：

```
>> A=rand(3)
A =
    0.8998    0.8180    0.2897
    0.8216    0.6602    0.3412
    0.6449    0.3420    0.5341
>> [Q,R]=qr(A)
Q =
   -0.6527    0.5439    0.5274
   -0.5960    0.0613   -0.8007
   -0.4678   -0.8369    0.2842
R =
   -1.3786   -1.0873   -0.6423
         0    0.1992   -0.2685
         0         0    0.0314
>> Q'*Q
ans =
    1.0000   -0.0000   -0.0000
   -0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000
>> Q*R
ans =
    0.8998    0.8180    0.2897
    0.8216    0.6602    0.3412
    0.6449    0.3420    0.5341
```



```
>> B=rand(2,4)
B =
    0.6946    0.7948    0.5226    0.1730
    0.6213    0.9568    0.8801    0.9797
>> [QQ,RR]=qr(B)
QQ =
   -0.7453   -0.6667
   -0.6667    0.7453
RR =
   -0.9319   -1.2303   -0.9763   -0.7821
         0    0.1832    0.3076    0.6149
>> QQ*QQ'
ans =
    1.0000    0.0000
    0.0000    1.0000
>> QQ*RR
ans =
    0.6946    0.7948    0.5226    0.1730
    0.6213    0.9568    0.8801    0.9797
```

4. SVD 分解（奇异值分解）

奇异值分解也是常用的矩阵分解之一。MATLAB 中通过 svd 函数实现矩阵的奇异值分解。

（1）*s*=svd[*A*]返回矩阵 ***A*** 的奇异值组成的列向量。

（2）[*U*,*S*,*V*]=svd(*A*)将矩阵 ***A*** 分解为三个矩阵的乘积，即 ***A*=*U**S*V'**；其中 ***U*** 和 ***V*** 是正交矩阵，***S*** 是一个对角矩阵，其对角元素为矩阵 ***A*** 奇异值的降序排列。

**例 14-10**　SVD 分解（奇异值分解）。

**解**：在命令窗口输入：

```
>> A=rand(3)
A =
    0.2714    0.7373    0.8939
    0.2523    0.1365    0.1991
    0.8757    0.0118    0.2987
>> s=svd(A)
s =
    1.3549
    0.7474
    0.0205
>> [U,S,V]=svd(A)
U =
   -0.8259    0.5422    0.1545
   -0.2505   -0.1073   -0.9622
   -0.5051   -0.8334    0.2244
S =
    1.3549         0         0
         0    0.7474         0
         0         0    0.0205
V =
   -0.5386   -0.8158   -0.2107
   -0.4791    0.5022   -0.7199
   -0.6931    0.2868    0.6613
>> U*S*V'
```

```
ans =
    0.2714    0.7373    0.8939
    0.2523    0.1365    0.1991
    0.8757    0.0118    0.2987
>> norm(A)
ans =
    1.3549
```

例 14-10 也验证了矩阵的 2-范数就是矩阵最大的奇异值。

5．Schur 分解

MATLAB 中实现 Schur 分解的命令是 schur。其语法格式为：

$$[U,T]=\text{schur}(A)$$

其中 $U$ 是一个正交矩阵，$T$ 是一个上三角矩阵，称为 $A$ 的 Schur 矩阵，并且满足 $A=U*T*U'$。

**例 14-11** Schur 分解。

**解：** 在命令窗口输入：

```
>> A=rand(3)
A =
    0.6614    0.0648    0.4235
    0.2844    0.9883    0.5155
    0.4692    0.5828    0.3340
>> [U,T]=schur(A)
U =
   -0.3417   -0.8710   -0.3530
   -0.7719    0.4744   -0.4232
   -0.5361   -0.1279    0.8344
T =
    1.4722    0.1789    0.1084
         0    0.6182   -0.1054
         0         0   -0.1067
>> U*T*U'
ans =
    0.6614    0.0648    0.4235
    0.2844    0.9883    0.5155
    0.4692    0.5828    0.3340
>> U*U'
ans =
    1.0000         0   -0.0000
         0    1.0000    0.0000
   -0.0000    0.0000    1.0000
```

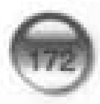
## 14.1.8 矩阵的对角元素操作

MATLAB 中支持的对矩阵对角元素的操作包括：提取对角元素，求矩阵对角元素的和（矩阵的迹），提取矩阵的上三角或下三角部分。

（1）函数 diag 提取矩阵的对角元素为一个向量。

（2）函数 trace 计算矩阵的迹。

（3）函数 tril 提取矩阵的下三角部分。





（4）函数 triu 提取矩阵的上三角部分。

实际上，tril（或 triu）还可以提取矩阵某条次对角线下方（或上方）的部分，tril(*A*,*k*)就是提取矩阵 ***A*** 的第 *k* 条对角线及其下方的部分。

**例 14-12**　对角元素操作。

**解**：在命令窗口输入：

```
>> A=rand(3)
A =
    0.6808    0.7942    0.0503
    0.4611    0.0592    0.4154
    0.5678    0.6029    0.3050
>> diag(A)
ans =
    0.6808
    0.0592
    0.3050
>> trace(A)
ans =
    1.0450
>> sum(diag(A))
ans =
    1.0450
>> triu(A)
ans =
    0.6808    0.7942    0.0503
         0    0.0592    0.4154
         0         0    0.3050
>> tril(A,1)
ans =
    0.6808    0.7942         0
    0.4611    0.0592    0.4154
    0.5678    0.6029    0.3050
```

### 14.1.9　矩阵分析函数总结

本小节对 14.1.1 到 14.1.8 的矩阵分析函数作个简单的列表总结，如表 14-1 所示。

表 14-1　矩阵分析函数

| 功能类别 | 函　数 | 说　明 |
|---|---|---|
| 表征矩阵特性的值、向量的求解 | det | 求矩阵的行列式（方阵） |
| | inv | 求矩阵的逆（方阵） |
| | pinv | 求矩阵的伪逆 |
| | rank | 求矩阵的秩 |
| | norm | 求矩阵的范数（方阵） |
| | cond | 求矩阵的条件数（方阵） |
| | eig | 求矩阵的特征值和特征向量（方阵） |
| | poly | 求矩阵的特征多项式（方阵） |
| | orth | 求矩阵空间的标准正交基 |

续表

| 功能类别 | 函数 | 说明 |
|---|---|---|
| 矩阵分解 | lu | 矩阵的 LU 分解 |
| | chol | 矩阵的 Cholesky 分解 |
| | qr | 矩阵的 QR 分解 |
| | svd | 矩阵的 SVD 奇异值分解 |
| | schur | 矩阵的 Schur 分解 |
| 对角元素操作 | diag | 提取矩阵的对角线元素 |
| | trace | 计算矩阵的迹 |
| | tril | 提取矩阵的下三角部分 |
| | triu | 提取矩阵的上三角部分 |

## 14.2 线性方程组

求解线性方程组是线性代数中的一个古老又基本的问题。本节内容介绍 MATLAB 中求解线性方程组的解决方案。

### 14.2.1 线性方程组的表示和种类

一般的线性方程组通常表示为如下的形式：

$$\begin{cases} a_{11}*x_1+a_{12}*x_2+\cdots+a_{1n}*x_n=b_1 \\ a_{21}*x_1+a_{22}*x_2+\cdots+a_{2n}*x_n=b_2 \\ \cdots \\ a_{m1}*x_1+a_{m2}*x_2+\cdots+a_{mn}*x_n=b_m \end{cases}$$

这里 $x_1,x_2,\cdots,x_n$ 是 $n$ 个未知数，$m$ 行代表了未知数满足的 $m$ 个方程。

这一线性方程组实际上可以用矩阵乘法表示为 $A*X=b$ 这样的形式，其中 $X$ 为 $n$ 个未知数构成的列向量$[x_1,x_2,\cdots,x_n]'$，$b$ 为方程组右边的数值构成的列向量$[b_1,b_2,\cdots,b_n]'$。$A$ 为方程组左边各个未知数的系数构成的矩阵，即：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$A$ 被称为线性方程组的系数矩阵，$[A\ b]$称为线性方程组的扩展矩阵。MATLAB 中，就是通过矩阵 $A$，列向量 $b$ 来描述一个线性方程组的。

根据系数矩阵 $A$ 的形状和秩，线性方程组可以分为：恰定方程组、欠定方程组和超定方程组：

（1）当方程个数等于未知数个数，即系数矩阵 $A$ 是一个方阵而且满秩时，即 rank($A$) $=m=n$，此线性方程组具有惟一解，称为恰定方程组。





（2）当方程个数小于未知数个数，即 $m<n$ 时，线性方程组有无穷多个可能的解，称为欠定方程组

（3）当有效方程个数（线性无关的方程个数）大于未知数个数时（一般 $m>n$ 都对应这种情况），线性方程组肯定没有精确解，称为超定方程组。

### 14.2.2　线性方程组的 MATLAB 求解

求解线性方程组有多种方法，在 MATLAB 中可以通过高斯消元法、矩阵除法、矩阵求逆等方法求解。

#### 1．高斯消元法

求解线性方程组可以通过高斯消元法，即将扩展矩阵[***A b***]通过行方向的线性运算变形为[***D nb***]形式，其中 ***D*** 和 ***A*** 尺寸相同且其最左上部分是一个单位矩阵。

（1）对于恰定方程组，***D*** 是和 ***A*** 同尺寸的单位矩阵，此时方程组的解的每一个分量可以通过 $X(i)=nb(i)/D(i,i)$ 计算得到；

（2）对于欠定方程组，***D*** 的左上部是一个比 ***A*** 小的 $k*k$ 的单位矩阵，这时候把 $i>k$ 的 $X(i)$置为 0，$i<=k$ 的 $X(i)$就可以通过 $X(i)=nb(i)/D(i,i)$ 计算得到；

（3）对于超定方程组，高斯消元后只能看出该方程组没有解。

MATLAB 中可以通过 rref([*A b*])，得到线性方程组扩展矩阵[*A b*]的高斯消元后的[*D nb*]矩阵，由此矩阵形式就可以直接计算出线性方程组的一般解。

**例 14-13**　高斯消元法求解恰定线性方程组。

**解：**在命令窗口输入：

```
>> A=rand(3)
A =
    0.2722    0.7468    0.4660
    0.1988    0.4451    0.4186
    0.0153    0.9318    0.8462
>> b=rand(3,1)
b =
    0.5252
    0.2026
    0.6721
>> D=rref([A b])
D =
    1.0000         0         0   -0.5704
         0    1.0000         0    1.3074
         0         0    1.0000   -0.6350
```

例 14-13 中待求解的线性方程组为 ***A*X=b***，通过 ***D***=rref([***A b***])计算，得到了此线性方程组扩展矩阵经高斯消元后的矩阵 ***D***。从矩阵 ***D*** 可以直接得到此方程组为恰定方程组，其解为 ***X***=[ −0.5704 1.3074 −0.6350]'。

**例 14-14**　高斯消元法求解欠定方程组。

**解：**在命令窗口输入：

```
>> A=rand(2,4)
A =
    0.7095    0.3046    0.1934    0.3028
    0.4289    0.1897    0.6822    0.5417
>> b=rand(2,1)
b =
    0.1509
    0.6979
>> rref([A b])
ans =
    1.0000         0  -43.8144  -27.5441  -47.1033
         0    1.0000  102.6812   65.1456  110.2017
```

例 14-14 中的线性方程组 $A*X=b$ 有 4 个未知数，2 个方程，因此必然是一个欠定方程组，通过 rref([*A b*])得到的高斯消元结果，可以看到此方程组至少有一个特解 $X$=[−47.1033 110.2017 0 0]'。

对于有无穷多个解的欠定方程，要获得其一般解的形式，可以先求线性方程组 $A*X=0$ 的解 $x_0$，再求 $A*X=b$ 的一个特解 $s$，那么 $A*X=b$ 的一般解就可以表示为 $s+r*x_0$。MATLAB 中求解 $A*X=0$，可以用 null 命令，null(*A*)返回 $A*X=0$ 的解空间的一组标准正交基。

**例 14-15** 欠定方程组的一般解。

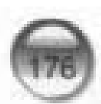
**解**：在命令窗口输入：

```
>> A=rand(2,4)
A =
    0.8180    0.3420    0.3412    0.7271
    0.6602    0.2897    0.5341    0.3093
>> b=rand(2,1)
b =
    0.8385
    0.5681
>> null(A)
ans =
   -0.2046   -0.6798
   -0.7783    0.5485
    0.4529    0.3424
    0.3837    0.3461
>> rref([A,b])
ans =
    1.0000         0   -7.4743    9.3574    4.3417
         0    1.0000   18.8759  -20.2561   -7.9331
```

例 14-15 中，通过 null 函数得到了 $A*X=0$ 的解空间的一组标准正交基，通过 rref ([*Ab*]) 可以得到 $A*X=b$ 的一个特解[4.3417 −7.9331 0 0]'。因此，欠定方程组 $A*X=b$ 的一般解可以表示为：

$$X=\begin{bmatrix}4.3417\\-7.9331\\0\\0\end{bmatrix}+k_1*\begin{bmatrix}-0.2046\\-0.7783\\0.4529\\0.3837\end{bmatrix}+k_2*\begin{bmatrix}-0.6798\\0.5485\\0.3424\\0.3461\end{bmatrix}$$

对于超定方程组，高斯消元结果只能看出该方程组没有一般意义的解。通常，各种数





学软件都会在最小二乘意义上给出超定方程组的解，MATLAB 中也是如此，后面会介绍。

2. 矩阵除法求解

求解线性方程组最简单的办法是用矩阵的除法。$A*X=b$ 的解可以由 $X=A\backslash b$ 得到，$X*A=b$ 的解可以有 $X=b/A$ 得到，注意，这里应用左除和右除对应的方程组形式的不同。

矩阵除法求解线性方程组，不会返回方程组类型的信息，即无论哪种类型的方程组，MATLAB 矩阵除法都会返回一个计算结果。对于恰定方程，这个结果就是其唯一解；对于欠定方程，此结果是其一个特解；对于超定方程，计算结果是方程组最小二乘意义上的解。

**例 14-16** 矩阵除法求解线性方程组。

**解**：在命令窗口输入：

```
>> A=rand(3);a=rand(3,1);
>> A\a
ans =
    0.0701
    0.8649
   -0.0342
>> rref([A a])
ans =
    1.0000         0         0    0.0701
         0    1.0000         0    0.8649
         0         0    1.0000   -0.0342
>> B=rand(2,3);b=rand(2,1);
>> B\b
ans =
    1.1966
         0
   -0.1830
>> rref([B b])
ans =
    1.0000         0  -38.3997    8.2232
         0    1.0000   51.2762   -9.3827
>> C=rand(4,3);c=rand(4,1);
>> x=C\c
x =
    1.1628
    0.7307
   -1.2451
>> b
b =
    0.7446
    0.2679
    0.4399
    0.9334
>> C*x
ans =
    0.8151
    0.2977
    0.4029
    0.8746
>> plot(C*x,'ro')
```

```
>> hold on;
>> plot(b,'b*') %对C*x和b两组向量数据描点绘图，如图14-1所示
```

例14-16中，**A*X=a** 是一个恰定方程组，**A\a** 和 rref([**A a**])得到的方程组的解一致。**B*X=b** 是一个欠定方方程组，这从 **B** 的形状以及 rref([**B b**])的结果都可以看到，**B\b** 返回方程组的一个特解，根据 rref([**B b**])也只能得到方程组的一个特解；**C*X=c** 是一个超定矩阵，通过 **C\c** 求得的解 ***x***，是最小二乘意义上方程组的解，因此 **C*x** 实际上不精确等于 ***c***，这从 plot 描绘的两组数据点的分布（如图14-1所示，红色圆圈代表 **C*x** 数据点，蓝色星号代表 ***b*** 数据点）也可以看到。

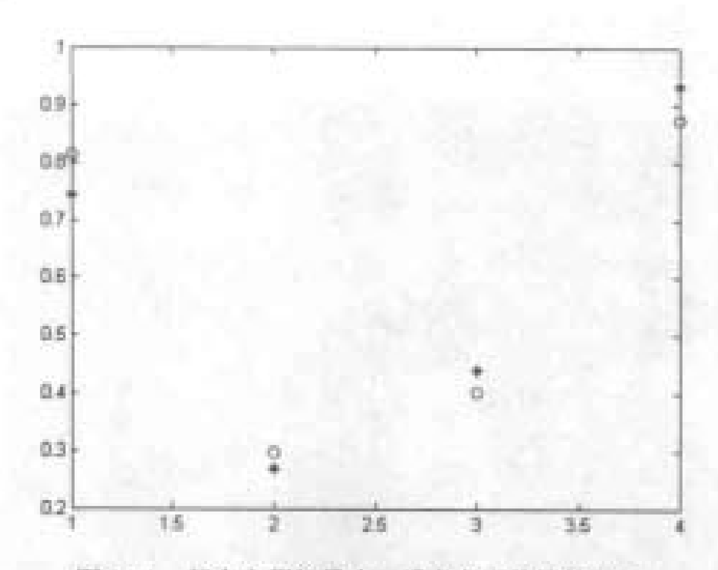


图14-1　超定方程组最小二乘意义下的解的图示

### 3. 矩阵求逆求解

求解线性方程组也可以通过逆矩阵的方法。对于方程组 **A*X=b**：

（1）当 ***A*** 是方阵时，**X=inv(A)*b**；

（2）当 ***A*** 不是方阵时，**X=pinv(A)*b**。

这样求解得到的线性方程组的解和高斯消元法、矩阵除法得到的结果应该是一致的。

MATLAB 求解线性方程组，效率最高的是矩阵除法，因为除法会自动识别系数矩阵 ***A*** 的特征而采用针对性的高效算法，因此，建议尽量多用矩阵除法求解线性方程组。

**例14-17**　矩阵求逆求解线性方程组。

**解**：在命令窗口输入：

```
>> A=rand(3);a=rand(3,1);
>> A\a
ans =
   -0.7571
    0.8344
    0.4630
>> inv(A)*a
ans =
   -0.7571
    0.8344
```



```
    0.4630
>> B=rand(4,3);b=rand(4,1);
>> B\b
ans =
    1.6192
    0.5701
   -5.2544
>> pinv(B)*b
ans =
    1.6192
    0.5701
   -5.2544
```

通过例 14-17 可以看出，无论对恰定方程组 ***A**X=a** 还是超定方程组 ***B**X=b**，矩阵求逆得到的方程组的解和矩阵除法得到的解是完全一致的。

## 14.3 特殊矩阵

线性代数中经常用到一些特殊的矩阵，MATLAB 提供了很多函数可以生成这些特殊矩阵，如表 14-2 所示。

表 14-2 特殊矩阵生产函数

| 函 数 | 说 明 |
|---|---|
| compan | 生成伴随矩阵 |
| gallery | 生成测试矩阵 |
| hilb | 生成严重病态的 Hilbert 矩阵 |
| invhilb | 生成逆 Hilbert 矩阵 |
| magic | 生成魔方矩阵 |
| pascal | 生成对称正定的 Pascal 矩阵 |
| rosser | 生成 Rosser 矩阵，用于测试特征值算法 |
| wilkinson | 生成 Wilkinson 矩阵，用于测试特征值算法 |
| hadamard | 生成 Hadamard 矩阵 |
| hankel | 生成 Hankel 矩阵 |
| vander | 生成 Vandermonde 矩阵 |
| toeplitz | 生成 Toeplitz 矩阵 |

## 14.4 稀疏矩阵

### 14.4.1 稀疏矩阵的存储方式

通常情况下，MATLAB 中顺序存储普通矩阵的各个元素。但有些应用领域，矩阵中零元素非常多，在庞大的矩阵中只有非常有限的非零元素，这时候如果还用顺序存储方式的话，会耗费很多不必要的内存空间，而且对于矩阵运算、处理也带来了不必要的复杂性。

针对这种情况，MATLAB 中可以采用稀疏矩阵提高存储、运算效率。

MATLAB 存储稀疏矩阵时，只存储矩阵中的非零元素及其位置索引。这对于大规模的零元素居多的数组存储是非常高效的。

实际上，MATLAB 通过三个数组来存储一个 *m* 行 *n* 列的稀疏矩阵的信息：

（1）第一个数组是浮点类型，存储了稀疏矩阵中所有非零元素的值（例如 *k* 个）；

（2）第二个数组是整型，存储了这 *k* 个非零元素的行索引；

（3）第三个数组是整型，存储了原来的 *m* 行 *n* 列元素中每一列第一个非零元素在前两个数组中的位置，以及第 *n* 列最后一个非零元素在前两个数组中的位置。

例如 *m* 行 *n* 列的系数矩阵中第 1 列只有第 3 行、第 5 行元素非零，第 2 列只有第 7 行元素非零，……，那么第二个数组前几个元素就是 3，5，7，……，第三个数组的前几个元素为 1，3，4，……

可以比较一下，用普通方式存储这个矩阵占用的内存空间是 8**m***n* 字节，而用稀疏矩阵存储占用的空间只有 8**k*+4*(*k*+*n*+1)，考虑到 *k*<<*m***n*，因此用稀疏矩阵存储是非常节省存储空间的。

### 14.4.2 稀疏矩阵的创建

MATLAB 中可以通过 sparse 函数把普通矩阵转换成稀疏矩阵，也可以通过 full 函数把稀疏矩阵转换为对应的普通矩阵。

**例 14-18** 稀疏矩阵的创建。

**解**：在命令窗口输入：

```
>> A=(rand(20)>0.99);
>> S=sparse(A)
S =
   (7,1)        1
   (7,3)        1
   (8,12)       1
  (13,13)       1
   (8,15)       1
  (20,18)       1
>> whos
  Name      Size                    Bytes  Class

  A         20x20                     400  logical array
  S         20x20                     114  logical array (sparse)

Grand total is 406 elements using 514 bytes
```

在例 14-18 中，通过 whos 显示当前工作区中的普通矩阵 ***A*** 和相应的稀疏矩阵 ***S***，可以看到它们占用的内存空间有明显的差别。***A*** 是一个 20 行 20 列的逻辑数组，每个元素占 1 个字节，因此共占 1*20*20=400 字节；***A*** 中只有 6 个非零元素，因此 ***S*** 占用 1*6+4*(6+20+1)=114 字节。



### 14.4.3　稀疏矩阵函数

表 14-3 列出了 MATLAB 中操作稀疏矩阵时最常用的函数。

表 14-3　稀疏矩阵函数

| 函　数 | 说　明 |
|---|---|
| sparse | 把普通矩阵转换为稀疏矩阵 |
| full | 把稀疏矩阵转换为普通矩阵 |
| nnz | 统计矩阵中的非零元素个数 |
| nzmax | 统计保存矩阵中非零元素占用的存储单元数目 |
| spalloc | 给稀疏矩阵分配内存空间 |
| issparse | 测试是否属于稀疏矩阵 |
| spy | 图形化显示稀疏矩阵非零元素分布 |

**例 14-19**　稀疏矩阵函数应用。

**解：** 在命令窗口输入：

```
>> S=sparse(randn(100)>3)
S =
   (88,4)       1
   (10,7)       1
   (41,10)      1
    (2,17)      1
   (54,23)      1
   (99,23)      1
   (18,33)      1
   (14,57)      1
   (45,58)      1
   (63,65)      1
   (31,66)      1
   (11,80)      1
   (57,93)      1
   (95,93)      1
>> nnz(S)
ans =
    14
>> nzmax(S)
ans =
    14
>> issparse(S)
ans =
     1
>> spy(S)          %显示结果如图 14-2 所示
```

程序运行后，输出如图 14-2 所示的结果。

通过例 14-19，可以看到，一般情况下 nnz 和 nzmax 返回结果一致，这表示一般情况下稀疏矩阵存储中的第一个数组都用来存储非零元素。但要注意，当稀疏矩阵 *S* 是由其他稀疏矩阵运算得到时，通常 nnz(*S*)都会显著小于 nzmax(*S*)，见例 14-20。

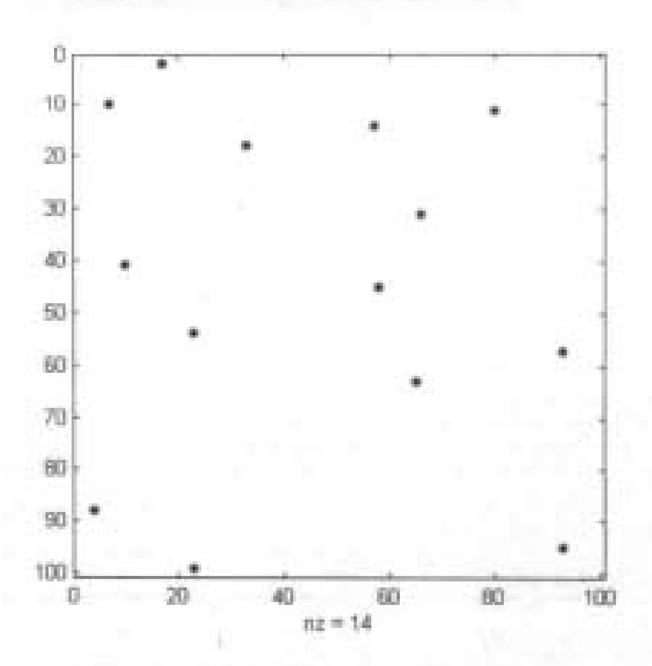



图 14-2　稀疏矩阵非零元素图形显示结果

**例 14-20**　nnz 和 nzmax 区别。

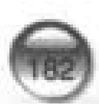
**解**：在命令窗口输入：

```
>> S=sparse(rand(100)>0.98);
>> nnz(S)
ans =
   197
>> nzmax(S)
ans =
   197
>> B=double(S);
>> SS=B*B;
>> nnz(SS)
ans =
   352
>> nzmax(SS)
ans =
   420
>> whos
  Name      Size                    Bytes  Class

  B         100x100                  2768  double array (sparse)
  S         100x100                  1389  logical array (sparse)
  SS        100x100                  5444  double array (sparse)
  ans         1x1                       8  double array

Grand total is 815 elements using 9609 bytes
```

从例 14-20 可以清楚地看到，对于由 sparse 函数直接创建的稀疏矩阵 ***S***，其 nnz 值等于 nzmax 值，而对于通过稀疏矩阵运算得到的稀疏矩阵 SS，其 nnz 值小于 nzmax 值，这是因为 SS 生成时分配了更多的存储空间准备用于存储其非零元素。





## 14.5 小结

本章讲解了 MATLAB 中处理线性代数的基本元素——矩阵和解决线性代数的基本问题——线性方程组的方法。其中讲解的关于矩阵的各种表征值、分解方法等，在线性代数及各种工程分析领域都有重要和广泛的应用，读者应该熟练掌握。

本章最后讲解了 MATLAB 中针对大规模多零元素的矩阵的存储、运算解决方案——稀疏矩阵。读者应该理解稀疏矩阵及其在 MATLAB 中的存储方式，对于各种稀疏矩阵的操作函数，读者也应该有所了解。



# 第 15 章 数据分析

MATLAB 中提供了丰富的函数和图形界面工具用于数据分析，包括数据绘图、统计量计算、相关性分析、插值、滤波和傅里叶分析等。另外，MATLAB 还有专门的统计工具箱可以进行各种专业的数据统计分析。

本章重点讲解基础统计分析和线性回归，并对傅里叶分析和 MATLAB 统计工具箱作简单介绍。

## 15.1 数据分析概述和数据预处理

### 15.1.1 数据分析概述

数据分析是指对测量或者采样得到的数据利用一定的分析方法获取其特征的过程，在信号和图像处理、滤波分析等很多领域，数据分析都有重要的应用。

数据分析根据要获取的数据特征不同，有很多不同的方法，如简单统计分析、数据拟合、傅里叶分析等。

MATLAB 提供了丰富的函数和图形界面工具用于数据分析。

MATLAB 中的数据分析函数，大多可以同时接受一维数组和二维数组形式的输入参数：

（1）一般地，当参数为一维数组（向量）时，分析函数以这组数据为数据样本进行相应的分析；

（2）当参数为二维数组（矩阵）时，分析函数把每一列当做一个数据样本进行分析。后续的各节中会对各种数据分析函数进行应用实例。



表 15-1 列出了 MATLAB 中图形界面的分析工具和其功能说明。

表 15-1 MATLAB 中图形界面的分析工具及功能

| 工 具 | 功 能 |
|---|---|
| 绘图 | 对工作区中的变量绘图显示 |
| 数据统计工具 | 计算数据集的各种统计量 |
| 基本拟合工具 | 用多项式模型或样条模型对数据进行拟合，并对拟合结果和残差进行可视化显示 |
| 时序信号分析工具 | 分析以时间为索引的数据并图形化显示分析结果 |

另外，MATLAB 还有众多工具箱可以用于各种专业领域的数据分析。这包括生物信息学工具箱、曲线拟合工具箱、样条工具箱、神经网络工具箱、图像处理工具箱、信号处理工具箱和统计工具箱等。

## 15.1.2 数据导入

数据分析中遇到的第一个问题就是数据的导入。一般用于数据分析的数据规模都比较大，在利用 MATLAB 进行数据分析之前，需要先把这些数据导入到 MATLAB 工作区，MATLAB 提供了数据导入向导，使得这一过程变得十分容易。

单击 MATLAB 主界面下 File 菜单的 Import Data 子菜单，就会打开导入向导窗口（如图 15-1 所示），按照窗口提示进行操作，就可以很方便地把本地文件中的数据导入到 MATLAB 工作区中。

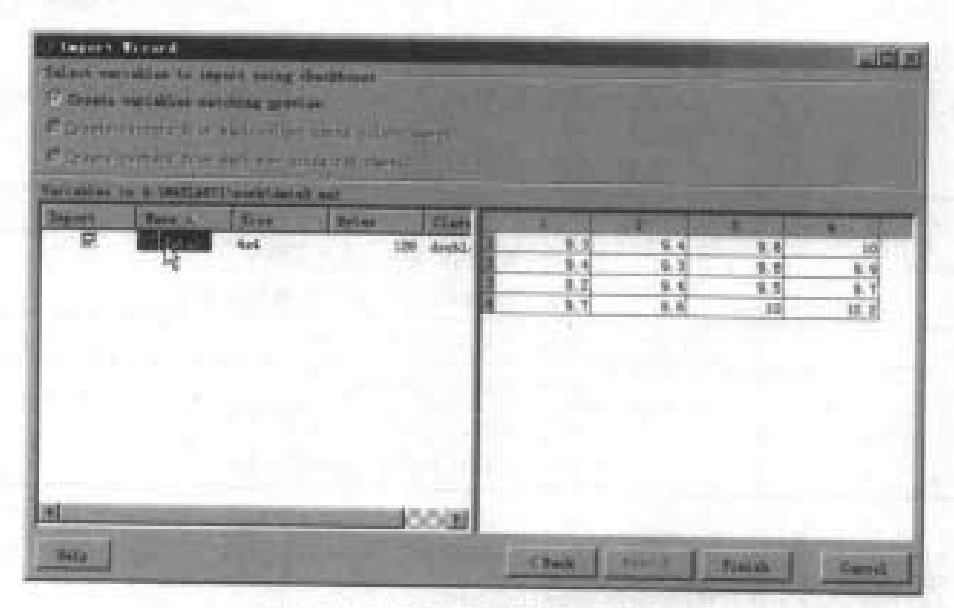


图 15-1 MATLAB 数据导入向导窗口

许多图形界面的分析工具也有数据导入向导，操作类似，就不赘述。

另外，MATLAB 提供了 load 函数可以从 ASCII 格式的.dat 文件或者 MATLAB 自带格式的.mat 文件中导入数据到工作区中。load 函数接受的参数比较复杂，后面章节还会进行详细介绍，读者也可参考 MATLAB 的帮助文件。

### 15.1.3 遗失数据的处理

数据分析中经常遇到原始数据不全，出现部分遗失的情况。怎样处理遗失的数据才能不影响分析结果的客观性，是一个比较困难的问题。一般有两种处理方法：

（1）忽略该数据项；

（2）通过插值方法估计遗失的数据项。插值估计的方法，请读者参考本书第 16 章。

这里介绍 MATLAB 用非数值量 NaN 表示遗失数据的处理方法。

NaN 参与的几乎所有的数学运算都返回 NaN，这使得用 NaN 表示的遗失数据在数据分析的结构中仍然以 NaN 形式出现。

**例 15-1** NaN 数据参与分析。

**解**：在命令窗口输入：

```
>> a = magic(3);
>> a(2,2) = NaN %用 NaN 表示遗失数据
a =
     8     1     6
     3   NaN     7
     4     9     2
>> sum(a)    %对数据集进行求和
ans =
    15   NaN    15
```

从例 15-1 可见，用 NaN 表示遗失数据后，有 NaN 的一组数据的求和结果仍然是 NaN。这样，有遗失数据的第二列数据集就能很容易地被找出来。

MATLAB 中删除具有 NaN 数据的数据集或计算结果的方法很多，如表 15-2 所示。

表 15-2 删除数据中的 NaN

| 方 法 | 说 明 |
|---|---|
| i=find(~isnan(x))<br>x=x(i) | 找出一维数组中非 NaN 数据的索引<br>用索引截取数组数据，从而删除 NaN 数据 |
| x=x(~isnan(x)) | 删除一维数组中的 NaN 数据 |
| x(isnan(x))=[] | 删除一维数组中的 NaN 数据 |
| X(any(isnan(X),2))=[]<br>X(any(isnan(X),1))=[] | 删除二维数组 X 中包含 NaN 的数据行<br>删除二维数组 X 中包含 NaN 的数据列 |

## 15.2 基础统计分析

基础的统计分析，是指通过计算数据集的最大最小值、平均值、中位值、出现频率最高值、方差和标准差等统计量来获知数据集的分布特性。

MATLAB 中可以通过相应的函数或数据统计工具完成这些分析。另外，MATLAB 还提供了统计工具箱可以进行更多专业的统计分析，其基础知识见本章第 15.5 节。



### 15.2.1　命令窗口统计分析

在 MATLAB 命令窗口中，可以通过调用多种统计分析函数进行数据的基础统计分析。MATLAB 中的基础统计分析函数如表 15-3 所示。

表 15-3　基础统计分析函数

| 函　数 | 说　明 |
|---|---|
| max | 返回数据集的最大值 |
| min | 返回数据集的最小值 |
| mean | 计算数据集的平均值 |
| median | 返回数据集的中位值 |
| mode | 返回数据集中出现频率最高的值 |
| var | 计算数据集的方差（用于测函数数据集中数据分布的离散程度） |
| std | 计算数据集的标准差（方差的算术平方根） |

当这些函数的输入参数是一维数组时，则以该一维数组为数据集进行统计计算。若输入参数是二维数组时，则以二维数组的每一列为一个数据集分别进行统计分析。

**例 15-2**　基础数据统计分析。

**解：**在命令窗口输入：

```
>> A=randn(100,5);         %产生 5 列随机数据集，每列包括 100 个数据
>> [mx,mxrow]=max(A)       %求各列数据的最大值和所在行号
mx =
    2.3726    1.8705    2.6903    2.1764    2.7316
mxrow =
    82    39    93    67    56
>> mmx=max(A(:))           %求 5 列数据的总体最大值
mmx =
    2.7316
>> mu=mean(A)              %求数据均值
mu =
   -0.0099   -0.1476   -0.0677    0.0572   -0.0543
>> sigma=std(A)        %求数据标准差
sigma =
    0.9977    0.9659    0.9777    0.8690    1.0061
>> varv=var(A)             %求数据方差
varv =
    0.9955    0.9329    0.9560    0.7552    1.0121
>> sqrt(varv)              %标准差等于方差的算术平方根
ans =
    0.9977    0.9659    0.9777    0.8690    1.0061
>> sum((A-repmat(mu,size(A,1),1)).^2)./(size(A,1)-1)    %用方差公式计算方差，
结果一致
ans =
    0.9955    0.9329    0.9560    0.7552    1.0121
```

### 15.2.2 MATLAB 数据统计工具

MATLAB 数据统计工具是一个图形用户界面的统计交互工具，利用它可以方便地得到数据集的统计量特征，并可视化地显示统计特征。

在 MATLAB 图像窗口中，单击打开 Tools 菜单下的 Data Statistics 子菜单，就会出现 MATLAB 数据统计窗口。MATLAB 统计工具中可以选择进行统计分析的数据集合，但需要先通过 plot 命令显示在绘图窗口中。MATLAB 统计分析结果给出数据集的最大最小值、平均值、中位值、标准差和数据分布范围。

**例 15-3** MATLAB 数据统计工具应用。

**解：** 在命令窗口输入：

```
>> A=rand(20,3);            %产生三列随机数据，每列 20 行
>> plot(A,'.')              %在图像窗口以实心点显示这些数据，如图 15-2 所示
>> legend('data1','data2','data3') %对三列数据添加标注
```

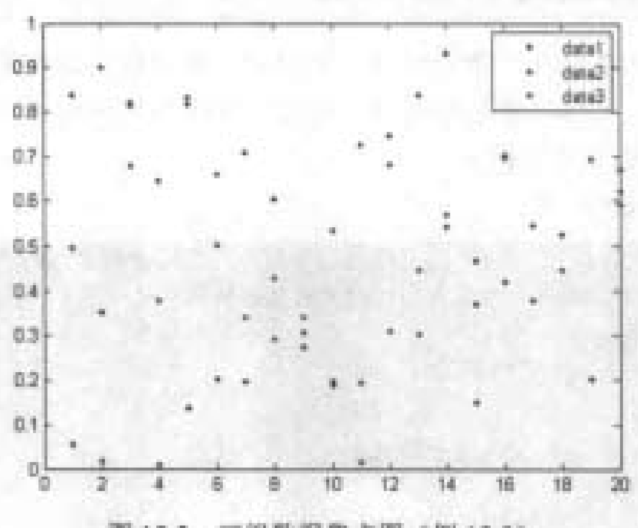



图 15-2 三组数据集点图（例 15-3）

打开图像窗口中的 Tools 菜单下的 Data Statistics 子菜单，弹出数据分析窗口，如图 15-3 所示，显示了数据分析的数据集（data 1 代表第一列数据），数据的最大最小值、平均值、中位值、标准差和数据范围。



图 15-3 数据统计工具窗口（例 15-3）



如图 15-3 所示，选中各项统计量后的复选框，则可以将它们显示到绘图窗口中，与图 15-3 对应的绘图窗口如图 15-4 所示。数据均值和标准差已经显示在绘图窗口中了。

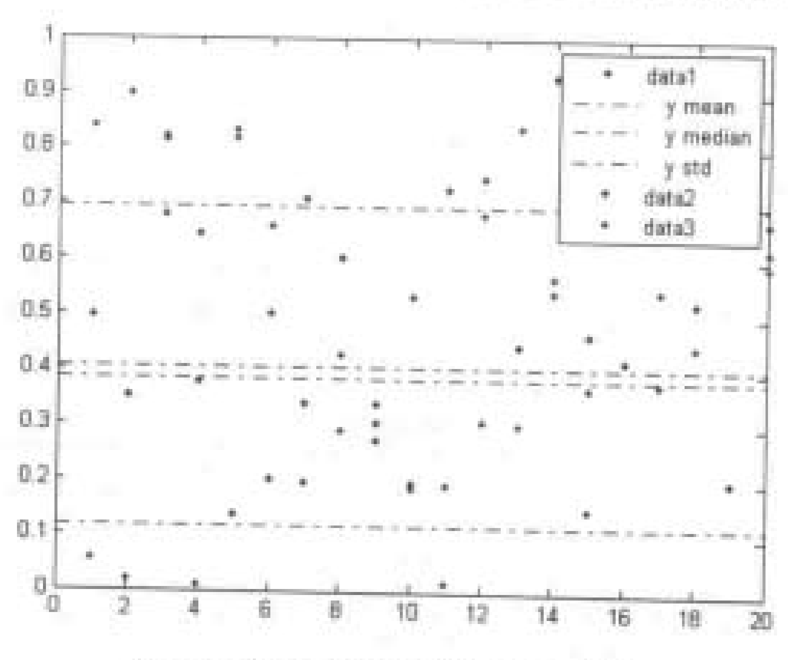



图 15-4　数据统计结果的图像显示（例 15-3）

单击数据统计窗口中的 Save to workspace…按钮，可以打开统计结果保存窗口（如图 15-5 所示），这时可以有选择地将统计结果返回保存到 MATLAB 工作区中。




图 15-5　统计结果保存窗口

统计结果以结构体的形式保存在 MATLAB 工作区。

**例 15-3（续）**　MATLAB 数据统计工具应用。

**解：** 在命令窗口输入：

```
Variables have been created in the current workspace.
Variables have been created in the current workspace.
>> xstats
xstats =
      min: 1
      max: 20
     mean: 10.5000
   median: 10.5000
      std: 5.9161
    range: 19
>> ystats
```

```
ystats =
       min: 0.0099
       max: 0.9318
      mean: 0.4059
    median: 0.3858
       std: 0.2877
     range: 0.9220
```

### 15.2.3 多组数据的相关分析

分析多组数据之间的相关性，也是数据统计分析的重要部分。描述数据相关性最重要的两个统计量是协方差和相关系数，它们刻画两组数据变化的协同性。MATLAB 中提供了计算两组数据间的协方差和相关系数的函数，如表 15-4 所示。

表 15-4 数据相关性分析函数

| 函 数 | 说 明 |
| --- | --- |
| cov(*x*) | 计算一维数组数据 *x* 方差 |
| cov(*x*,*y*) | 计算两组一维数组数据 *x*,*y* 的协方差 |
| cov(*X*) | 计算二维数组 *X* 的协方差矩阵 |
| corrcoef(*x*,*y*) | 计算两组数据 *x*,*y* 的相关系数 |
| corrcoef(*X*) | 计算二维数组 *X* 的相关系数矩阵 |

在计算协方差和相关系数时，这两个函数还是把每一列数组当做一个数据集。

二维数组 ***X*** 的协方差矩阵中，第 *i* 行 *i* 列的对角元素表示 ***X*** 第 *i* 列数据的方差；第 *i* 行 *j* 列的元素（和第 *j* 行 *i* 列元素）表示 ***X*** 第 *i* 列和第 *j* 列数据的协方差。

二维数组 ***X*** 的相关系数矩阵中，第 *i* 行 *i* 列的对角元素肯定为 1，表示第 *i* 列数据自身具有同步变化的线性相关性；第 *i* 行 *j* 列的元素（和第 *j* 行 *i* 列元素）表示 ***X*** 第 *i* 列和第 *j* 列数据的相关系数。

**例 15-4** 数据相关性分析。

**解**：在命令窗口输入：

```
>> x=rand(1,5);y=randn(1,5);
>> A=rand(3,4);
>> cov(x)
ans =
    0.1199
>> var(x)
ans =
    0.1199
>> cov(x,y)
ans =
    0.1199   -0.2187
   -0.2187    0.8526
>> cov(A)
ans =
    0.0082   -0.0161   -0.0051   -0.0182
   -0.0161    0.0316    0.0115    0.0320
   -0.0051    0.0115    0.0133   -0.0106
```



```
   -0.0182    0.0320   -0.0106    0.0870
>> corrcoef(x,y)
ans =
    1.0000   -0.6839
   -0.6839    1.0000
>> corrcoef(A)
ans =
    1.0000   -0.9960   -0.4869   -0.6784
   -0.9960    1.0000    0.5633    0.6098
   -0.4869    0.5633    1.0000   -0.3113
   -0.6784    0.6098   -0.3113    1.0000
```

## 15.3 用线性回归模型拟合数据

用线性回归模型对已知数据进行拟合分析，是数据处理中重要的分析方法，在数据拟合和预测方面是最基本最重要的方法。很多非线性拟合问题也可以转化成线性回归模型。MATLAB 中用最小二乘法进行线性回归模型拟合分析有两种方法：

（1）命令窗口下通过内部函数进行回归拟合；

（2）用基本拟合工具进行回归分析。

回归模型的好坏通常用数据观测值和模型估计值之间的残差分布来衡量。一个好的回归模型，残差应该是接近随机分布的。

MATLAB 中的基本拟合工具可以计算并图形化显示各数据点处的残差分布。另外，MATLAB 的曲线拟合工具箱还提供了更多专业的拟合方法。

### 15.3.1 命令窗口下的线性回归

线性回归模型可以表示为：

$$y = a_0 + a_1 * f_1(x) + a_2 * f_2(x) + \cdots + a_n * f_n(x)$$

其中 $f_1(x), f_2(x), \cdots, f_n(x)$ 是可以通过自变量 $x$ 计算得到的，$a_0, a_1, a_2 \cdots, a_n$ 是待拟合的系数，它们在模型中都是线性形式的。

最常用的线性回归是多项式函数回归，即 $f_1(x), f_2(x), \cdots, f_n(x)$ 是 $x$ 的幂函数。

MATLAB 中用多项式拟合函数 polyfit 可以对已知数据进行指定阶的多项式回归。本书中第 17 章对 polyfit 有详细的讲解，这里简单举例说明一下该函数的用法和结果的意义。

**例 15-5** 多项式回归。

**解：** 在命令窗口输入：

```
>> t = [0 .3 .8 1.1 1.6 2.3]';
>> y = [0.5 0.82 1.14 1.25 1.35 1.40]';      %产生已知数据
>> p=polyfit(t,y,2)
p =
   -0.2387    0.9191    0.5318
>> t2 = 0:.1:2.8;
>> y2=polyval(p,t2);     %计算拟合函数在更多点上的值
>> plot(t,y,'o',t2,y2)          %画出原来数据点和拟合曲线，如图 15-6 所示
```

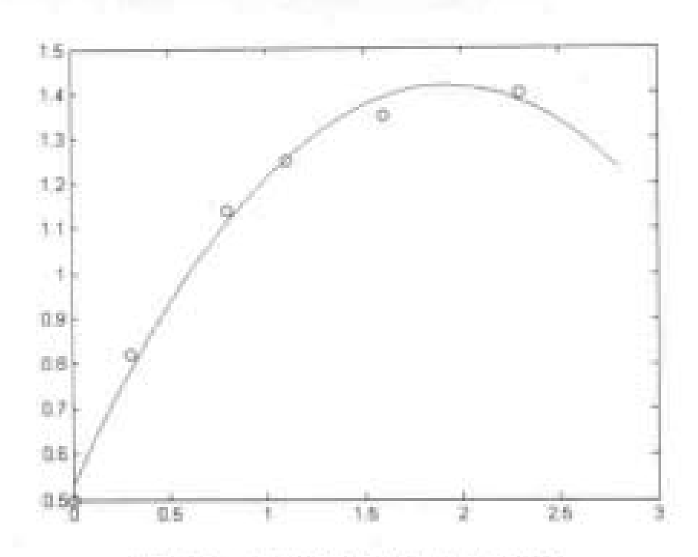


图 15-6 多项式拟合结果（例 15-5）

对于一般的线性回归模型

$$y = a_0 + a_1 * f_1(x) + a_2 * f_2(x) + \cdots + a_n * f_n(x)$$

MATLAB 可以用构造模型矩阵，然后用矩阵除法求得拟合系数。

实际上，该模型用数组形式可以表示为：

$$\begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 & f_1(x_1) & f_2(x_1) & \cdots & f_n(x_1) \\ x_2 & f_1(x_2) & f_2(x_2) & \cdots & f_n(x_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_n & f_1(x_n) & f_2(x_n) & \cdots & f_n(x_n) \end{pmatrix} * \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \cdots \\ a_n \end{pmatrix}$$

若简写成

$$\boldsymbol{Y} = \boldsymbol{X} * \boldsymbol{A}$$

则系数矩阵 $\boldsymbol{A}$ 可以用矩阵除法表示为

$$\boldsymbol{A} = \boldsymbol{X}^{-1} * \boldsymbol{Y} = \boldsymbol{X} \backslash \boldsymbol{Y}$$

MATLAB 中通过矩阵除法可以求得拟合参数，而这一结果也是最小二乘意义的。

**例 15-6** 一般线性回归（数组除法）。

**解：** 在命令窗口输入：

```
>> t = [0 .3 .8 1.1 1.6 2.3]';
>> X = [ones(size(t))  exp(-t)  t.*exp(-t)];
>> y = [0.5 0.82 1.14 1.25 1.35 1.40]';
>> a = X\y
a =
    1.3974
   -0.8988
    0.4097
>> T = (0:0.1:2.5)';
>> Y = [ones(size(T))  exp(-T)  T.*exp(-T)]*a;
>> plot(T,Y,'-',t,y,'o')     %如图 15-7 所示
```

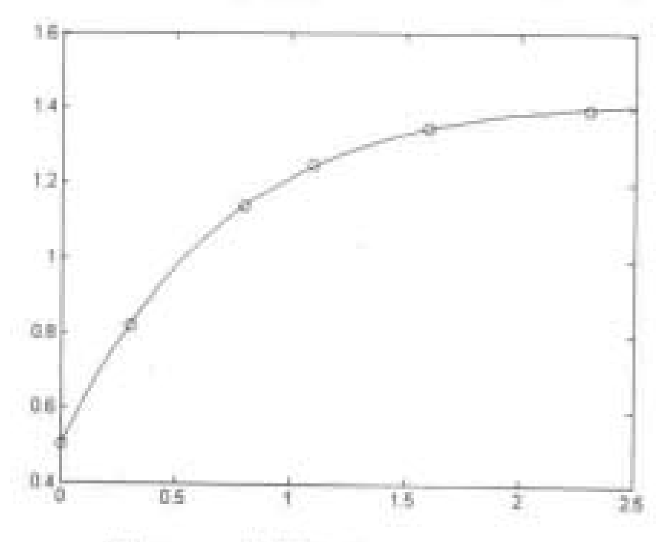


图 15-7 一般线性回归结果（例 15-6）

例 15-6 的结果模型可以表示为：

$$y = 1.3974 - 0.8900 * e^{-t} + 0.4097 * t * e^{-t}$$

同样的，用矩阵除法也可以进行多元线性模型回归分析，只需要改动相应的系数矩阵即可，如例 15-7 所示。

**例 15-7** 多元线性回归。

**解**：在命令窗口输入：

```
>> x1 = [.2 .5 .6 .8 1.0 1.1]';
>> x2 = [.1 .3 .4 .9 1.1 1.4]';
>> y  = [.17 .26 .28 .23 .27 .24]';
>> X = [ones(size(x1))  x1  x2];
>> a = X\y
a =
   0.1018
   0.4844
  -0.2847
```

例 15-7 中的多元线性模型可表示为：

$$y = 0.1018 + 0.4844x_1 - 0.2847x_2$$

### 15.3.2 用基本拟合工具进行回归分析

MATLAB 中除了可以用内部函数在命令窗口下进行回归分析外，还可以通过图形用户界面的基本拟合工具进行数据回归。

**例 15-8** 应用基本拟合工具进行线性回归分析。

**解**：在命令窗口输入：

```
>> t = [0 .3 .8 1.1 1.6 2.3]';
>> y = [0.5 0.82 1.14 1.25 1.35 1.40]';
>> plot(t,y,'o')        %原始数据描点如图 15-8 所示
```

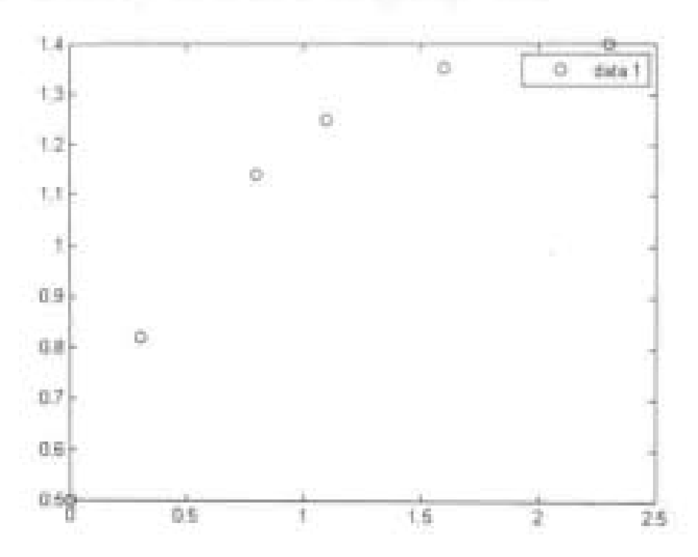



图 15-8 应用基本拟合工具线性回归的原始数据点（例 15-8）

打开图像窗口中的 Tools 菜单下的 Basic fitting 子菜单，弹出基本拟合窗口，如图 15-9 所示。



图 15-9 MATLAB 基本拟合窗口（例 15-8）

图 15-9 所示的窗口下可以选择待拟合的数据集（已经通过 plot 命令传递到绘图窗口）、中心化和归一化数据（在选择高阶多项式回归时有用，避免出现病态的模型矩阵）、选择拟合方式（简单线性、二次、高次多项式模型或样条模型），还可以选择在图像中显示拟合函数和残差分布图。

这时候，绘图窗口中会画出线性和二次回归函数，并以子图方式画出拟合中的残差分布，如图 15-10 所示。在图 15-9 所示的基本拟合窗口中进行不同的选择，绘图窗口会同步显示不同的拟合结果和残差分布。



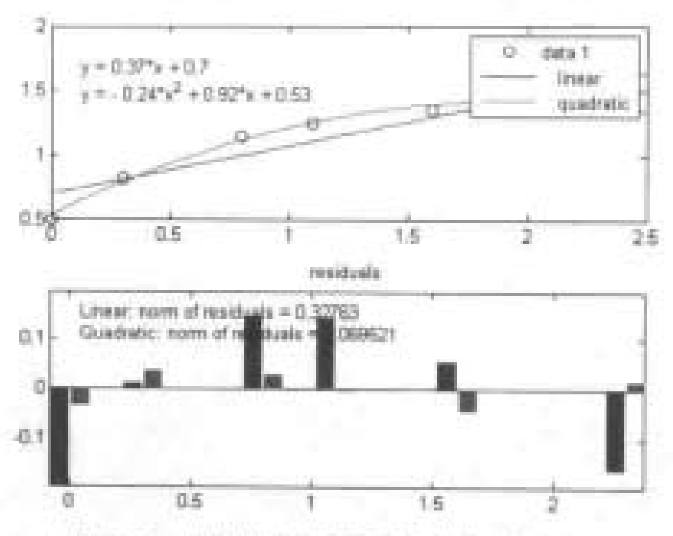



图 15-10　拟合结果显示和残差分布（例 15-8）

在基本拟合工具中还可以获得拟合参数，单击图 15-9 中红色圆圈处的扩展箭头，基本拟合窗口中会出现数据拟合的结果，包括拟合模型、参数计算结果和残差。再单击扩展箭头则可以计算拟合函数在指定点的函数值（如图 15-11 所示）。

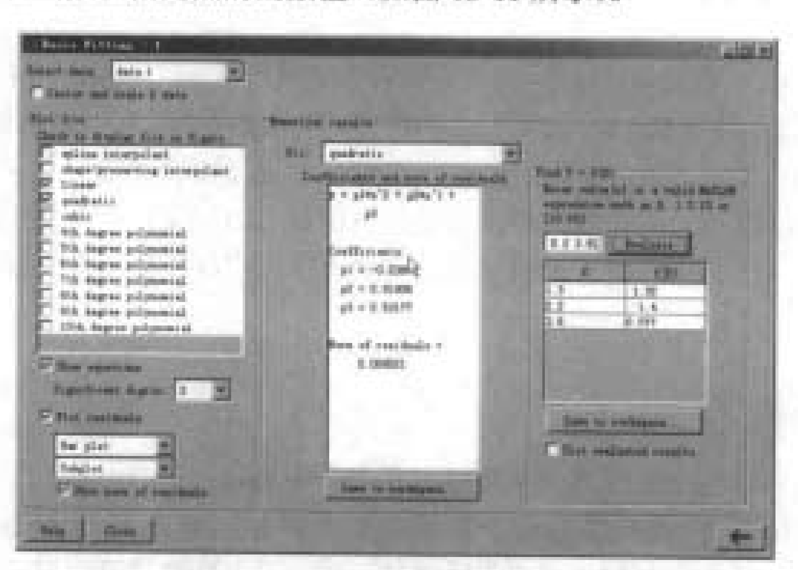


图 15-11　扩展后的基本拟合窗口（例 15-8）

单击窗口上的 Save to workspace…按钮，可以把拟合参数的计算值和指定点的函数值保存到 MATLAB 工作区中指定名字的变量中（如图 15-12 所示）。

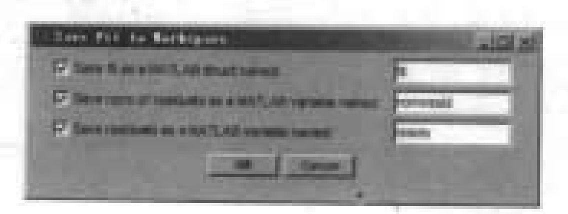


图 15-12　拟合结果保存窗口（例 15-8）



这时候可以在 MATLAB 工作区中调用这些拟合结果。

例 15-8（续）　应用基本拟合工具进行线性回归分析。

解：在命令窗口输入：

```
Variables have been created in the current workspace.
>> fit
fit =
     type: 'polynomial degree 2'
    coeff: [-0.2387 0.9191 0.5318]
>> normresid
normresid =
    0.0695
>> resids
resids =
   -0.0318
    0.0340
    0.0257
   -0.0039
   -0.0412
    0.0171
```

## 15.4　其他分析方法初步

本节简单介绍一下 MATLAB 中的有限差分和傅里叶分析，其中傅里叶分析经常用在信号处理过程中，这里只对 MATLAB 中与这两种分析方法相关的部分函数做简单的实例讲解，更专业和详细的知识，请用户参阅其他专著或 MATLAB 的联机帮助。

此外，MATLAB 中还提供了时序信号分析方法和 GUI 工具，以及专业的信号分析工具箱，通过这些专业工具可以进行滤波分析、谱分析等，这属于相应专业分析领域的范畴，本书不再赘述。

### 15.4.1　有限差分

MATLAB 中提供了 diff 函数可以对离散的数据系列进行差分分析。

diff($x$)计算向量 $x$ 的相邻的两个数据之间的插值，返回向量[$x(2)-x(1)$ $x(3)-x(2)$ ⋯ $x(n)-x(n-1)$]。diff 函数主要配合其他逻辑测试函数，用于检测某组数据点中是否有相邻的等值元素、是否单调递增、是否等差等。常用的几种用法如表 15-5 所示。

表 15-5　有限差分分析

| 函　　数 | 意义说明 |
|---|---|
| any(diff(x)==0) | 数据系列 x 中是否有相邻的等值数据点 |
| all(diff(x)>0) | 数据系列 x 是否单调递增 |
| all(diff(diff(x))==0) | 数据系列 x 是否等差 |

例 15-9　有限差分分析。

解：在命令窗口输入：



```
>> x=ones(1,5)
x =
     1     1     1     1     1
>> any(diff(x)==0)
ans =
     1
>> y=logspace(1,2,5)
y =
   10.0000   17.7828   31.6228   56.2341  100.0000
>> all(diff(y)>0)
ans =
     1
>> z=linspace(1,10,5)
z =
    1.0000    3.2500    5.5000    7.7500   10.0000
>> all(diff(diff(z))==0)
ans =
     1
```

### 15.4.2　傅里叶分析初步

傅里叶变换能够把一个离散的采样信号分解成一组不同频率的正弦信号的叠加，这在信号分析中有重要的应用。MATLAB 中采用快速傅里叶变换算法进行离散傅里叶变换。表 15-6 列出了 MATLAB 中傅里叶分析的常用函数。

表 15-6　MATLAB 中傅里叶分析的函数

| 函　数 | 说　明 |
|---|---|
| fft | 一维傅里叶变换 |
| fft2 | 二维傅里叶变换 |
| fftn | N 维傅里叶变换 |
| ifft | 一维傅里叶逆变换 |
| ifft2 | 二维傅里叶逆变换 |
| ifftn | N 维傅里叶逆变换 |
| abs | 计算复数的幅值 |
| angle | 计算复数的相位角 |
| unwrap | 通过增减若干个 2*pi 使相邻相位角差值不超过 pi |

**例 15-10**　傅里叶分析。

**解**：在命令窗口输入：

```
>> t=1:20;
>> y=sin(0.6*t)+cos(3*t);
>> plot(t,y)      %绘制 y 关于 t 的函数图像，如图 15-13 所示
>> fy=fft(y)      %对 y 进行快速傅里叶变换
fy =
  Columns 1 through 4
  -1.0029              -1.1124 + 0.4953i   2.2906 - 9.8863i  -0.5144 - 1.4108i
  Columns 5 through 8
  -0.6258 - 1.2124i  -0.6610 - 1.3196i  -0.6764 - 1.6024i  -0.6830 - 2.1392i
  Columns 9 through 12
  -0.6821 - 3.3042i  -0.6472 - 7.8870i  -0.8805              -0.6472 + 7.8870i
  Columns 13 through 16
```

```
  -0.6821 + 3.3042i  -0.6830 + 2.1392i  -0.6764 + 1.6024i  -0.6610 + 1.3196i
  Columns 17 through 20
  -0.6258 + 1.2124i  -0.5144 + 1.4108i   2.2906 + 9.8863i  -1.1124 - 0.4953i
>> mfy=abs(fy); %求y经傅里叶变换后的幅值
>> plot(t,mfy)  %如图15-14所示
>> afy=angle(fy);   %求y经傅里叶变换后的相位角
>> uafy=unwrap(afy);     %调整间距超过pi的相位角
>> plot(t,afy,'r*',t,afy,'r')   %用红色*标记数据点并连线画图
>> hold on  %打开图形的叠加绘制模式
>> plot(t,uafy,'go',t,uafy,'g')%用绿色o标记数据点并连线画图，如图15-15所示
>> hold off %关闭图形的叠加绘制模式
>> ifft(fy)-y
ans =
  1.0e-015 *
  Columns 1 through 8
  -0.0555   -0.4441    0.0833   -0.2220    0.1110   -0.1665         0 0.2220
  Columns 9 through 16
   0.4441    0.1110   -0.0555   -0.2220   -0.4441    0.3331    0.2220       0
  Columns 17 through 20
   0.0208         0   -0.0902    0.2220
```

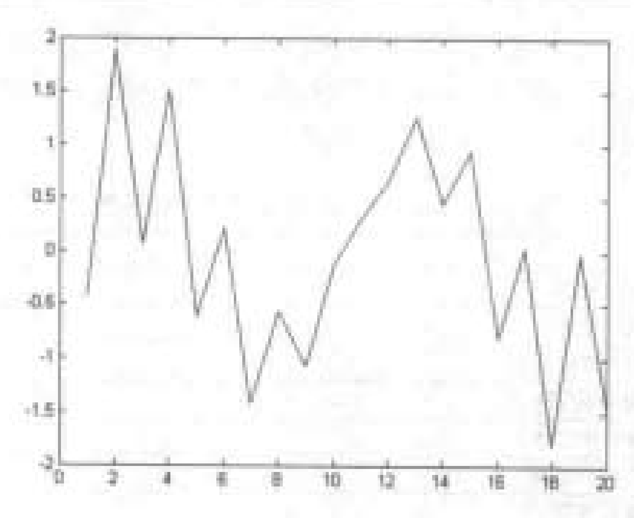



图15-13　原始数据点绘图（例15-10）

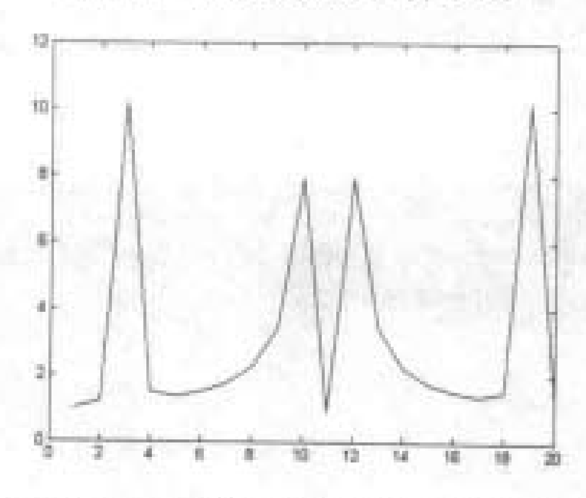



图15-14　傅里叶变换后的幅值绘图（例15-10）

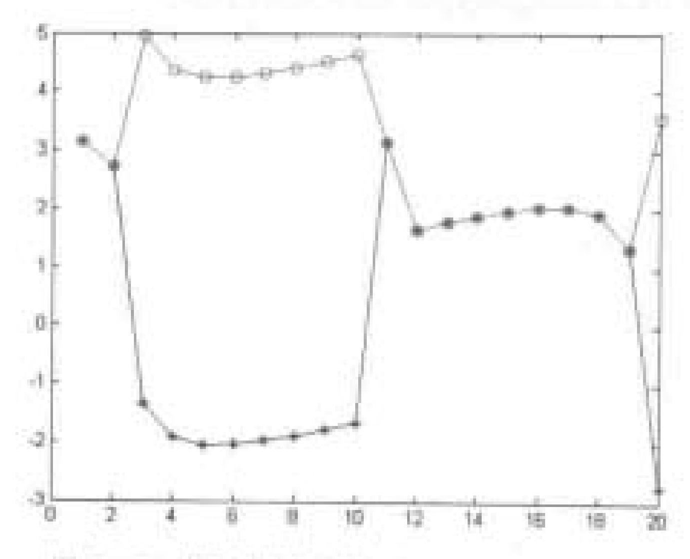



图 15-15　傅里叶变换后的相位角绘图（例 15-10）

## 15.5　MATLAB 统计工具箱初步

本节介绍 MATLAB 统计工具箱中用途最广泛的一些基础知识，包括概率密度函数、分布函数、逆分布函数和随机数的产生。统计工具箱中还包括线性模型和非线性模型回归、多元统计、假设检验、实验设计、统计绘图和隐马尔科夫模型等内容，有相应需要的读者请自行参考 MATLAB 联机帮助。

### 15.5.1　概率密度函数

MATLAB 中计算某种概率分布在指定点的概率密度的函数，都以代表特定概率分布的字母开头，以 pdf（probability density function）结尾，如表 15-7 所示。

表 15-7　概率密度函数

| 函数语法 | 说　明 |
|---|---|
| unidpdf(*X*,*N*) | 计算 1 到 *N* 上的离散均匀分布在 *X* 每一点处的概率密度 |
| poisspdf(*X*,LAMBDA) | 计算参数为 LAMBDA 的泊松分布在 *X* 每一点处的概率密度 |
| unifpdf(*X*,*A*,*B*) | 计算 U(*A*,*B*)连续均匀分布在 *X* 每一点处的概率密度 |
| exppdf(*X*, mu) | 计算参数为 mu 的指数分布在 *X* 每一点处的概率密度 |
| normpdf(*X*,mu,sigma) | 计算正态分布 N(mu,sigma)在 *X* 每一点处的概率密度 |
| tpdf(*X*,*V*) | 计算 t 分布 t(*V*)在 *X* 每一点处的概率密度 |
| chi2pdf(*X*,*V*) | 计算 $x^2$ 分布 $X^2(V)$在 *X* 每一点处的概率密度 |
| fpdf(*X*,*V1*,*V2*) | 计算 F 分布 F(*V1*,*V2*)在 *X* 每一点处的概率密度 |

**例 15-11**　概率密度函数。

**解：**在命令窗口输入：

```
>> a=3.2;
>> ay=normpdf(a,2,2)    %计算 N(2,2)正态分布在 3.2 处的概率密度值
ay =
   0.1666
>> x=-2:0.2:6;
>> y=normpdf(x,2,2);
>> plot(a,ay,'*',x,y)   %画 N(2,2)的概率密度函数并标出计算点位置
>> grid on   %打开图形网格，如图 15-16 所示
```

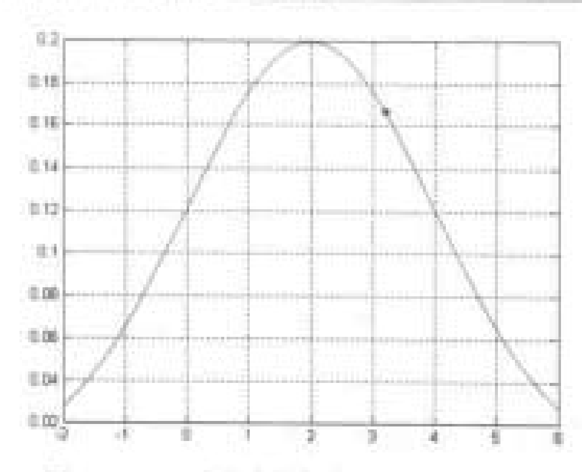



图 15-16　正态分布的概率密度（例 15-11）

## 15.5.2　概率分布函数

概率分布函数用于计算某种概率分布在某个区间上的累积概率，它是在指定区间上对概率密度函数进行积分。

MATLAB 中计算概率密度的函数都是从负无穷为积分下界。MATLAB 中计算某种概率分布在指定点的累积概率的函数，都以代表特定概率分布的字母开头，以 cdf（cumulative distribution function）结尾，如表 15-8 所示。

表 15-8　概率分布函数

| 函数语法 | 说　明 |
| --- | --- |
| unidcdf(*X*,*N*) | 计算 1 到 *N* 上离散均匀分布在 1 到 *X* 上每一点处的累积概率 |
| poisscdf(*X*,LAMBDA) | 计算参数为 LAMBDA 的泊松分布在 *X* 每一点处的累积概率 |
| unifcdf(*X*,*A*,*B*) | 计算 U(*A*,*B*)连续均匀分布在 *X* 每一点处的累积概率 |
| expcdf(*X*, mu) | 计算参数为 mu 的指数分布在 *X* 每一点处的累积概率 |
| normcdf(*X*,mu,sigma) | 计算 N(mu,sigma)正态分布在 *X* 每一点处的累积概率 |
| tcdf(*X*,*V*) | 计算 t(*V*)的 t 分布在 *X* 每一点处的累积概率 |
| chi2cdf(*X*,*V*) | 计算$\chi^2(V)$的$\chi^2$分布在 *X* 每一点处的累积概率 |
| fcdf(*X*,*V*1,*V*2) | 计算 F(*V*1,*V*2)的 F 分布在 *X* 每一点处的累积概率 |

**例 15-12**　概率分布函数。

**解：**在命令窗口输入：

```
>> x=-4:4;
```







```
>> y=unidcdf(x,10)
y =
  Columns 1 through 8
       0         0         0         0         0    0.1000    0.2000    0.3000
  Column 9
    0.4000
>> tx=-5:0.01:12;
>> plot(tx,ty,x,y,'*')  %绘制离散均匀分布的分布函数图并标注计算点，如图 15-17 所示
```

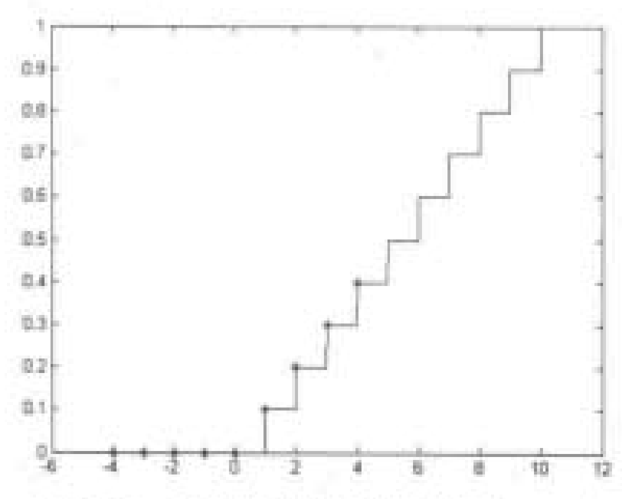




图 15-17　概率分布函数图和计算点（例 15-12）

### 15.5.3　逆概率分布函数

逆概率分布函数是概率分布函数的逆函数，用来计算特定的概率分布从负无穷到哪个点的累积概率能达到指定值。

MATLAB 中的逆概率分布函数，都以代表特定概率分布的字母开头，以 inv（inverse cumulative distribution function）结尾。因为累积概率是 0 到 1 之间的非负数值，所以给定的概率值必须是此范围内的数，否则计算结果可能返回 NaN，如表 15-9 所示。

表 15-9　逆概率分布函数

| 函数语法 | 说　明 |
|---|---|
| unidinv(*P*,*N*) | 计算 1 到 *N* 上离散均匀分布在 *P* 的每一概率值对应的逆概率点 |
| poissinv(*P*,LAMBDA) | 计算参数为 LAMBDA 的泊松分布在 *P* 的每一概率值对应的逆概率点 |
| unifinv(*P*,*A*,*B*) | 计算 U(A,B)连续均匀分布在 *P* 的每一概率值对应的逆概率点 |
| expinv(*P*, mu) | 计算参数为 mu 的指数分布在 *P* 的每一概率值对应的逆概率点 |
| norminv(*P*,mu,sigma) | 计算正态分布 N(mu,sigma)在 *P* 的每一概率值对应的逆概率点 |
| tinv(*P*,*V*) | 计算 t 分布 t(*V*)在 *P* 的每一概率值对应的逆概率点 |
| chi2inv(*P*,*V*) | 计算$\chi^2$分布$\chi^2(V)$在 *P* 的每一概率值对应的逆概率点 |
| finv(*P*,*V*1,*V*2) | 计算 F 分布 F(*V*1,*V*2)在 *P* 的每一概率值对应的逆概率点 |

**例 15-13**　逆概率分布函数。

**解：** 在命令窗口输入：

```
>> p=0:0.2:0.9;
```

```
>> x=expinv(p,3)
x =
        0    0.6694    1.5325    2.7489    4.8283
>> tx=-2:0.01:10;
>> tp=expcdf(tx,3);
>> plot(tx,tp,x,p,'*')
>> grid on  %绘制指数分布的分布函数图并标注计算点，并打开网格，如图 15-18 所示
```

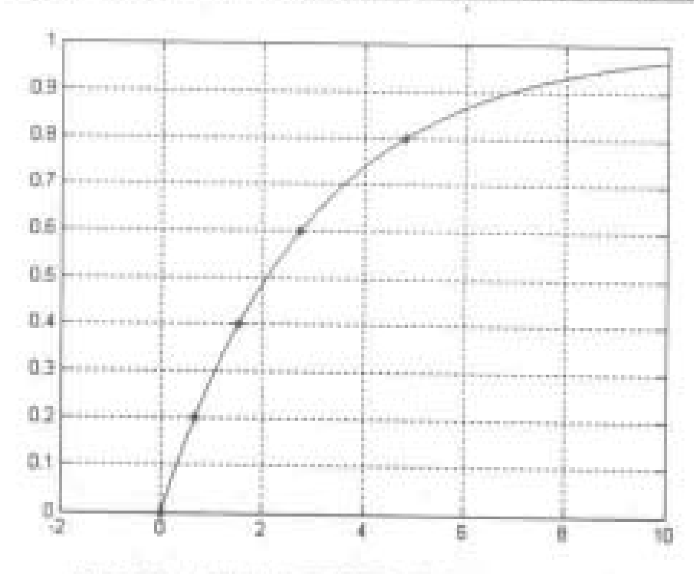



图 15-18　概率分布函数图和计算点（例 15-13）

### 15.5.4　随机数产生

MATLAB 中可以产生指定分布的随机数，相应的函数都是以代表特定概率分布的字母开头，以 rnd（random number）结尾，如表 15-10 所示。

表 15-10　随机数产生函数

| 函数语法 | 说　明 |
|---|---|
| unidrnd(*N*,*m*,*n*) | 产生 *m* 行 *n* 列个服从 1 到 *N* 上离散均匀分布的随机数 |
| poissrnd(LAMBDA,*m*,*n*) | 产生 *m* 行 *n* 列个服从参数为 LAMBDA 的泊松分布的随机数 |
| unifrnd(*A*,*B*,*m*,*n*) | 产生 *m* 行 *n* 列个服从 U(*A*,*B*)连续均匀分布的随机数 |
| exprnd( mu,*m*,*n*) | 产生 *m* 行 *n* 列个服从参数为 mu 的指数分布的随机数 |
| normrnd(mu,sigma,*m*,*n*) | 产生 *m* 行 *n* 列个服从正态分布 N(mu,sigma)的随机数 |
| trnd(*V*,*m*,*n*) | 产生 *m* 行 *n* 列个服从 t 分布 t(*V*)的随机数 |
| chi2rnd(*V*,*m*,*n*) | 产生 *m* 行 *n* 列个服从$\chi^2$分布$\chi^2$(*V*)的随机数 |
| frnd(*V*1,*V*2,*m*,*n*) | 产生 *m* 行 *n* 列个服从 F 分布 F(*V*1,*V*2)的随机数 |

**例 15-14**　随机数的产生。

**解：** 在命令窗口输入：

```
>> rd=normrnd(0,1,1,500);    %产生 500 个服从 N(0,1)正态分布的随机数
>> plot(rd,'o') %画出这些随机数点，并用基本统计工具进行分析，如图 15-19 所示
>> hist(rd) %绘制随机数的频率直方图
```





```
>> x=-5:0.2:5;
>> y=normpdf(x);
>> hold on
>> plot(x,y*500)        %叠加绘制理论上的频率图，如图15-20所示
```

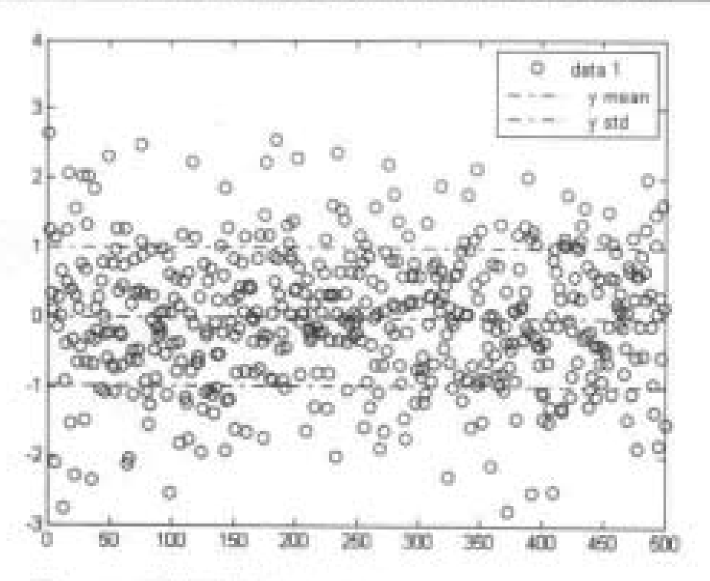



图 15-19　随机数描点和基本统计分析结果（例 15-14）

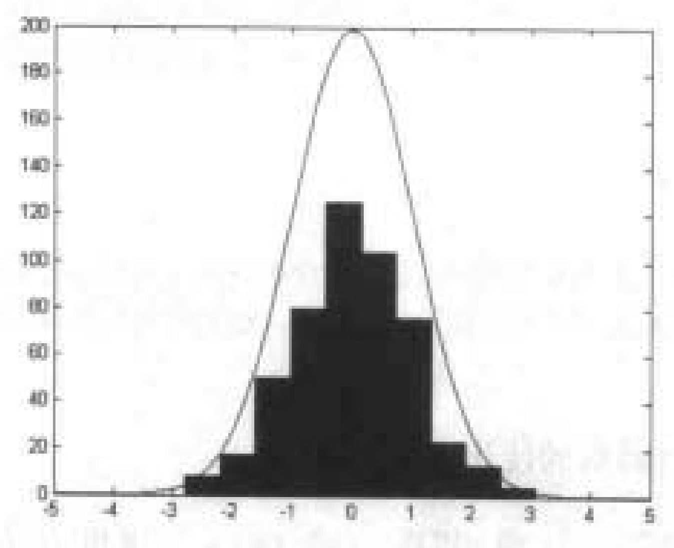


图 15-20　实际的频率直方图和理论的频率图（例 15-14）

## 15.6　小结

本章介绍了 MATLAB 中多种数据分析方法，其中基本统计分析和线性回归拟合是本章的重点内容，读者一定要熟练掌握相应的命令行分析和使用 GUI 工具交互分析的方法。

本章中介绍到的各种其他分析方法的基本知识，在许多领域都有广泛应用，读者应该简单地了解并能使用，而相应的专业领域的分析方法，如信号分析、图像分析等方面的知识，MATLAB 中提供了多个分析工具箱，供专业领域的读者选择学习。

# 第 16 章 数据插值

插值就是已知一组离散的数据点集，在集合内部某两个点之间预测函数值的方法。本章介绍 MATLAB 中一维插值、二维插值和高维插值的知识，重点讲解一维插值和二维插值，介绍多种插值算法各自的优缺点，并用可视化方法直观地比较各种算法插值的效果。

## 16.1 一维插值

插值运算是根据数据的分布规律，找到一个函数表达式可以连接已知的各点，并用这一函数表达式预测两点之间任意位置上的函数值。插值运算在信号处理和图像处理领域应用十分广泛。

### 16.1.1 一维插值函数的使用

若已知的数据集是平面上的一组离散点集（***x,y***），则其相应的插值就是一维插值。MATLAB 中一维插值的函数是 interp1，它有多种语法形式，如表 16-1 所示。

表 16-1 一维插值函数 interp1 的语法格式

| 语法形式 | 说　明 |
|---|---|
| *yi*=interp1(*x*,*Y*,*xi*) | 由已知点集（*x*,*Y*）插值计算 *xi* 上的函数值 *yi* |
| *yi*=interp1(*Y*,xi) | 相当于 *x*=1:length(*Y*)的 interp1(*x*,*Y*,*xi*) |
| *yi*=interp1(*x*,*Y*,*xi*,method) | 用指定插值方法计算插值点 *xi* 上的函数值 *yi* |
| *yi*=interp1(*x*,*Y*,*xi*,method,'extrap') | 对 *xi* 中超出已知点集的插值点用指定方法计算函数值 *yi* |
| *yi*=interp1(*x*,*Y*,*xi*,method,extrapval) | 用指定方法插值 *xi* 上的函数值 *yi*，*xi* 中超出已知点集处函数值取 extrapval |
| *pp*=interp1(*x*,*Y*,method,'pp') | 用指定方法进行插值，但返回结果为分段多项式 |



从表 16-1 知道，MATLAB 中一维插值有多种算法，由 interp1 函数中的 method 参数指定。MATLAB 中一维插值的各种算法如表 16-2 所示。

表 16-2　一维插值算法（method）

| method | 方法描述 |
|---|---|
| 'nearest' | 最邻近插值：插值点处函数值取与插值点最邻近的已知点上的函数值 |
| 'linear' | 分段线性插值：插值点处函数值由连接其最邻近的两侧点的线性函数预测，MATLAB 中 interp1 的默认方法 |
| 'spline' | 样条插值：默认为三次样条插值，可用 spline 函数代替 |
| 'pchip' | 三次 Hermite 多项式插值，可用 pchip 函数代替 |
| 'cubic' | 同 pchip，三次 Hermite 多项式插值 |
| 'v5cubic' | MATLAB 5 中的三次多项式插值，不能用于外插<br>当已知点不等间距时，转换为 spline 方法 |

表 16-2 的各种方法中：

（1）'nearest'方法速度最快，占用内存最小，但一般来说误差最大，插值结果最不光滑；

（2）'linear'分段线性插值方法则在速度和误差之间取得了比较好的均衡，其插值函数具有连续性，但在已知数据点处的斜率一般都会改变，因此不是光滑的，分段线性插值方法是 MATLAB 一维插值的默认方法；

（3）'spline'三次样条插值法是所有插值方法中运行耗时最长的，其插值函数以及插值函数的一阶、二阶导函数都连续，因此是最光滑的插值方法，占用内存上比'cubic'方法小，但当已知数据点不均匀分布时可能会出现异常结果；

（4）'cubic'三次多项式插值法中插值函数及其一阶导函数都是连续的，因此其插值结果也比较光滑，运算速度比'spline'方法略快，但占用内存最多。在实际的使用中，应根据实际需求和运算条件选择合适的算法。

另外，已知数据点不等间距分布时，interp1q 函数比 interp1 函数执行速度快，因为前者不检查已知数据点是否等间距，不过 interp1q 函数要求 $x$ 必须单调递增。

**例 16-1**　对 sin 函数进行分段线性一维插值。

**解**：在命令窗口输入：

```
>> x = 0:10;
>> y = sin(x);
>> xi = 0:.25:10;
>> yi = interp1(x,y,xi);
>> plot(x,y,'o',xi,yi)  %用 o 标出已知数据点，用直线顺次连接所有插值点，结果如图 16-1 所示
```

例 16-1 中用默认的方法（分段线性的 linear 方法）对已知的 10 个 sin 函数的数据点进行插值，用 plot 画出插值结果，从图 16-1 可以看出，分段线性插值就是用连接两个邻近的已知点的线性函数，插值计算该区间内插值点上的函数值。

**例 16-2**　其他几种方法对 sin 函数进行插值。

**解**：在命令窗口输入：

```
>> x = 0:10;
>> y = sin(x);
>> xi = 0:.25:10;
>> yi=sin(xi);
>> y1=interp1(x,y,xi,'nearest');
>> y2=interp1(x,y,xi,'spline');
>> y3=interp1(x,y,xi,'cubic');
>> plot(x,y,'o',xi,yi,xi,y1,'-.')    %最邻近插值，如图 16-2 所示
>> plot(x,y,'o',xi,yi,xi,y2,'-.')    %三次样条插值，如图 16-3 所示
>> plot(x,y,'o',xi,yi,xi,y3,'-.')    %三次多项式插值，如图 16-4 所示
```

图 16-1　sin 函数一维插值结果（例 16-1）

图 16-2　sin 函数最邻近插值（'nearest'）（例 16-2）

图 16-3　sin 函数三次样条插值（'spline'）（例 16-2）

图 16-4　sin 函数三次多项式插值（'cubic'）（例 16-2）

### 16.1.2　内插运算和外插运算

插值运算可以分为内插和外插两种：

（1）只对已知数据点集内部的点进行的插值运算称为内插，内插可以根据已知数据点的分布，构建能够代表分布特性的函数关系，比较准确地估测插值点上的函数值；

（2）当插值点落在已知数据集外部时的插值称为外插，要外插估计函数值是很难的。





MATLAB中没有指定外插算法时，对已知数据集外部点上函数值的估计都返回NaN。

interp1函数中可以通过添加'extrap'参数，指明插值算法也用于外插运算。另外还可以直接对数据集外的函数点赋值为extraval，一般赋值为NaN（MATLAB默认）或者0。

下面举几个例子简单说明一下外插可能出现的误差，以及表16-1已经列举过的MATLAB处理外插的几种方法。

**例16-3** 外插运算方法和误差。

**解：**在命令窗口输入：

```
>> x = 0:10;
>> y = sin(x);
>> xi=5:0.25:15;
>> yi=sin(xi);
>> y1=interp1(x,y,xi,'nearest')
y1 =
  Columns 1 through 8
   -0.9589   -0.9589   -0.2794   -0.2794   -0.2794   -0.2794    0.6570
0.6570
  Columns 9 through 16
    0.6570    0.6570    0.9894    0.9894    0.9894    0.9894    0.4121
0.4121
  Columns 17 through 24
    0.4121    0.4121   -0.5440   -0.5440   -0.5440       NaN       NaN
NaN
  Columns 25 through 32
       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
  Columns 33 through 40
       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
  Column 41
       NaN
>> y1=interp1(x,y,xi,'nearest','extrap');
>> y2=interp1(x,y,xi,'linear','extrap');
>> y3=interp1(x,y,xi,'spline','extrap');
>> y4=interp1(x,y,xi,'cubic','extrap');
>> y5=interp1(x,y,xi,'cubic',0);
>> plot(x,y,'o',xi,yi,xi,y1,xi,y2,xi,y3,xi,y4,xi,y5)    %各种外插结果，如图
16-5所示
>> legend('data','sin','nearest','linear','spline','cubic','0',2)  %在左上
方标注
>> table=[xi',yi',y1',y2',y3',y4',y5'];
>> n=size(table,1);
>> table([1:10,n-10:n],:)          %比较各种算法的内插和外插结果
ans =
    5.0000   -0.9589   -0.9589   -0.9589   -0.9589   -0.9589   -0.9589
    5.2500   -0.8589   -0.9589   -0.7890   -0.8578   -0.8897   -0.8897
    5.5000   -0.7055   -0.2794   -0.6192   -0.7032   -0.7176   -0.7176
    5.7500   -0.5083   -0.2794   -0.4493   -0.5065   -0.4963   -0.4963
    6.0000   -0.2794   -0.2794   -0.2794   -0.2794   -0.2794   -0.2794
    6.2500   -0.0332   -0.2794   -0.0453   -0.0342   -0.0454   -0.0454
    6.5000    0.2151    0.6570    0.1888    0.2142    0.2259    0.2259
    6.7500    0.4500    0.6570    0.4229    0.4498    0.4786    0.4786
    7.0000    0.6570    0.6570    0.6570    0.6570    0.6570    0.6570
    7.2500    0.8231    0.6570    0.7401    0.8215    0.7779    0.7779
   12.5000   -0.0663   -0.5440   -2.9344    0.0916   -3.5680         0
```

```
   12.7500    0.1826   -0.5440   -3.1734    0.6996   -3.7995         0
   13.0000    0.4202   -0.5440   -3.4124    1.4512   -4.0006         0
   13.2500    0.6316   -0.5440   -3.6515    2.3575   -4.1668         0
   13.5000    0.8038   -0.5440   -3.8905    3.4297   -4.2938         0
   13.7500    0.9260   -0.5440   -4.1295    4.6790   -4.3771         0
   14.0000    0.9906   -0.5440   -4.3686    6.1165   -4.4125         0
   14.2500    0.9936   -0.5440   -4.6076    7.7533   -4.3955         0
   14.5000    0.9349   -0.5440   -4.8466    9.6006   -4.3217         0
   14.7500    0.8180   -0.5440   -5.0857   11.6696   -4.1867         0
   15.0000    0.6503   -0.5440   -5.3247   13.9713   -3.9862         0
```

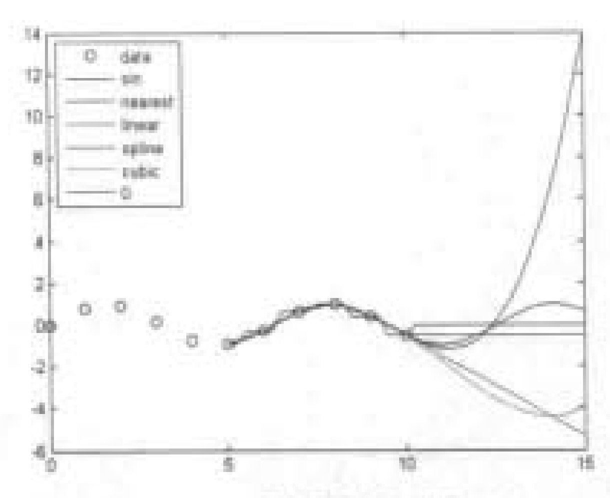



图 16-5　sin 函数外插结果（例 16-3）

从图 16-5 可以看出，不管用哪种插值方法，当插值点位于已知数据集合外时，插值运算对该处函数值的估计都很可能与实际函数值有很大的偏差。

表 16-2 中已经提到样条插值可以用 spline 函数，三次多项式插值可以用 pchip 函数。这两个函数的语法格式如表 16-3 所示。

表 16-3　spline 函数和 pchip 函数语法格式

| 函数语法 | 说　明 |
|---|---|
| yi=spline(x,y,xi) | 相当于 yi=interp1(x,y,xi,'spline') |
| pp=spline(x,y)<br>yi=ppval(pp,xi) | 返回分段样条插值函数<br>以 pp 为插值函数计算 xi 上的函数插值结果 |
| yi=pchip(x,y,xi) | 相当于 yi=interp1(x,y,xi,'cubic') |
| pp=pchip(x,y)<br>yi=ppval(pp,xi) | 返回分段三次 Hermite 多项式插值函数<br>以 pp 为插值函数计算 xi 上的函数插值结果 |

**例 16-4**　spline 函数和 pchip 函数。

**解：** 在命令窗口输入：

```
>> x = -3:3;
>> y = [-1 -1 -1 0 1 1 1];
>> t = -3:.01:3;
>> p = pchip(x,y,t);
>> ppol=spline(x,y)
ppol =
```



```
      form: 'pp'
    breaks: [-3 -2 -1 0 1 2 3]
     coefs: [6x4 double]
    pieces: 6
     order: 4
       dim: 1
>> s=ppval(ppol,t);
>> plot(x,y,'o',t,p,'-',t,s,'-.')  %画图，如图16-6所示
>> legend('data','pchip','spline','Location','SouthEast')  %在图像的右下方标注
```

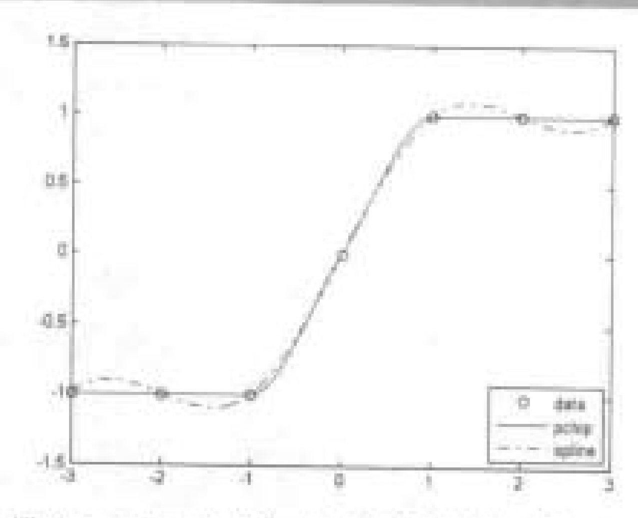



图16-6 spline函数和pchip函数插值结果（例16-4）

## 16.2 二维插值

已知点集是三维空间中的点的插值就是二维插值问题。二维插值问题在图像处理中有广泛的应用。

MATLAB中的二维插值函数是interp2，其用法类似于一维插值函数interp1。

interp2完整语法格式为：

*ZI*=interp2(*X*,*Y*,*Z*,*XI*,*YI*,method,extrapval)

表示在已知的（*X*,*Y*,*Z*）三维栅格点数据基础上，在（*XI*,*YI*）这些点上用method方法估计函数值，当插值点位于已知数据点集区域外时，函数值赋值为extrapval。

其中可选的方法有三种：'nearest'，'linear'和'cubic'，其意义说明如表16-4所示。

表16-4 二维插值方法

| method | 描 述 |
| --- | --- |
| 'nearest' | 最邻近点插值：<br>用最邻近插值点的已知点上的函数值估计插值点处的函数值。<br>最快速，占用内存最少，最粗糙 |
| 'linear' | 二维线性插值：（interp2默认方法）<br>用插值点最邻近的四个已知点上的函数值估计插值点处的函数值。<br>速度较快，内存占用较小，分片线性连续，不光滑 |

续表

| method | 描　述 |
| --- | --- |
| 'cubic' | 二维三次多项式插值；<br>用插值点最邻近的六个已知点上的函数值估计插值点处的函数值，<br>速度最慢，内存占用最多，最光滑 |

二维插值中已知数据点集（*X*,*Y*）必须是栅格格式，一般用 meshgrid 函数产生。interp2 函数要求（*X*,*Y*）必须是严格单调的，即单调递增或者单调递减。另外，若已知点集（*X*,*Y*）在平面上分布不是等间距时，interp2 函数首先通过一定的变换将其转换为等间距的。

当输入点集（*X*,*Y*）已经是等间距分布的话，可以在 method 参数前面加星号（*），即如'*cubic'这样输入参数，这样可以提高插值速度。

**例 16-5** 二维插值。

**解：**在命令窗口输入：

```
>> [X,Y] = meshgrid(-3:.25:3); %产生已知数据的栅格点
>> Z = peaks(X,Y);       %计算已知数据点上的函数值，peaks 是 MATLAB 的一个二元函数
>> [XI,YI] = meshgrid(-3:.125:3);       %产生更精细的插值栅格点
>> ZI = interp2(X,Y,Z,XI,YI);
>> mesh(X,Y,Z)  %绘制(X,Y,Z)已知数据的栅格图
>> hold on  %打开附加绘图模式，以后的绘图结果叠加在现有图像上
>> mesh(XI,YI,ZI+15)    %绘制(XI,YI,ZI+15)插值数据的栅格图(沿 Z 轴上移 15 个单位)
>> hold off %关闭附加绘图模式
>> axis([-3 3 -3 3 -5 20])      %调整坐标轴范围，最终结果如图 16-7 所示
```

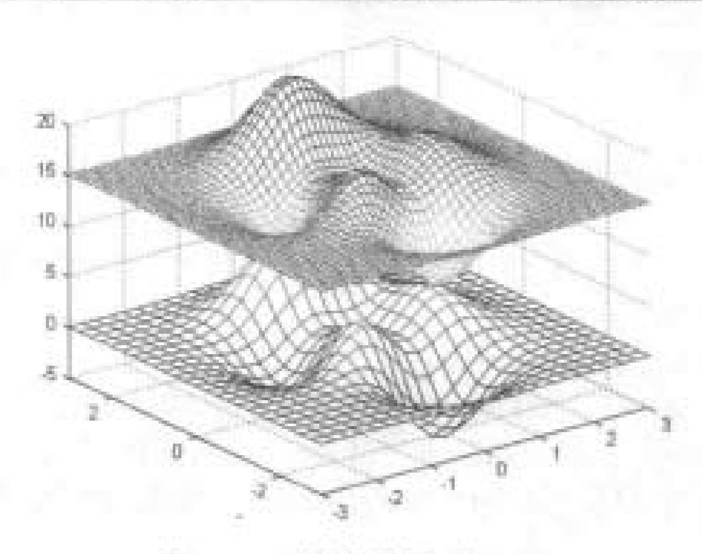



图 16-7　二维插值结果（例 16-5）

例 16-5 中对 MATLAB 内部的一个二元函数 peaks 进行了精细内插值，采用默认的二维线性算法，由于已知数据密集度较高，因此线性插值结果已经很光滑。

下面的例 16-6 通过绘制插值结果的三维表面图，直观地比较三种二维插值方法在插值效果上的差别。





**例 16-6** 二维插值方法效果比较。

**解：** 在命令窗口输入：

```
>> [x,y] = meshgrid(-3:1:3);      %产生已知数据栅格点
>> z = peaks(x,y);  %计算已知点上的函数值
>> surf(x,y,z)  %画基于已知数据点的三维表面图，如图 16-8 所示
>> title('graphic based on original data')      %对图形加标题
>> [xi,yi] = meshgrid(-3:0.25:3);  %产生更精细的插值栅格点
>> zi1 = interp2(x,y,z,xi,yi,'nearest');
>> surf(xi,yi,zi1)  %画基于最邻近法插值的三维表面图，如图 16-9 所示
>> title('nearest-interpolation method')
>> zi2 = interp2(x,y,z,xi,yi,'linear');
>> surf(xi,yi,zi2)        %画基于二维分段线性插值的三维表面图，如图 16-10 所示
>> title('linear-interpolation method')
>> zi3 = interp2(x,y,z,xi,yi,'cubic');
>> surf(xi,yi,zi3)        %画基于二维三次多项式插值的三维表面图，如图 16-11 所示
>> title('cubic-interpolation method')
```

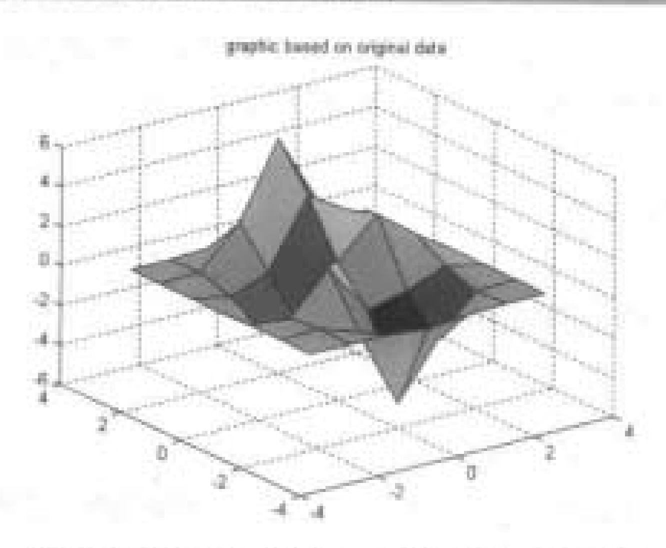



图 16-8 基于原始数据绘制的三维表面图（例 16-6）

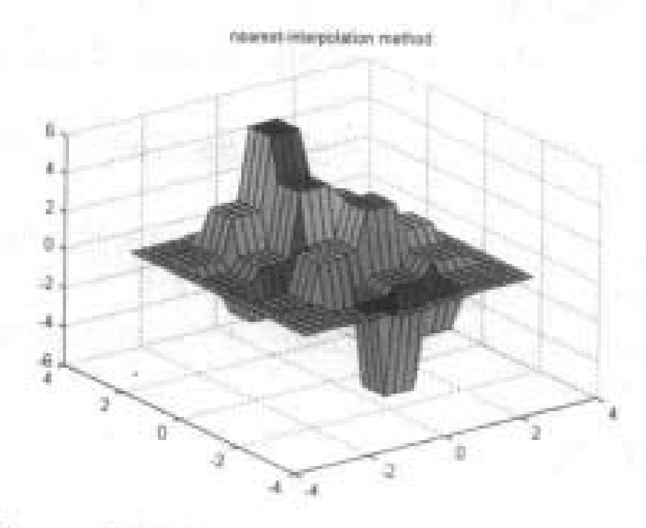



图 16-9 基于最邻近法插值数据的三维表面图（例 16-6）

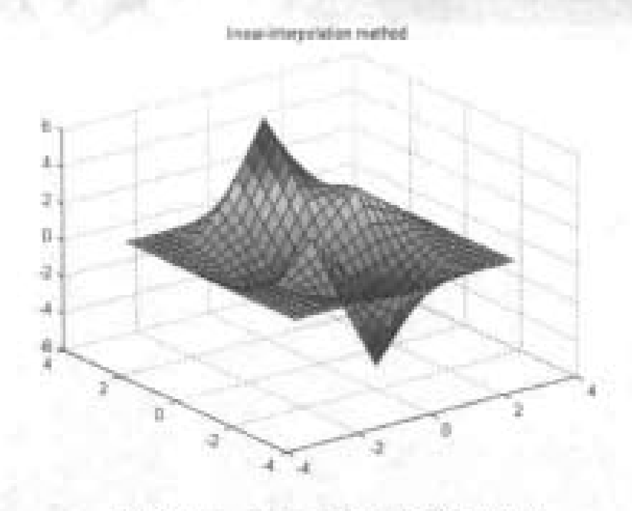



图 16-10 基于二维分段线性插值数据的三维表面图（例 16-6）

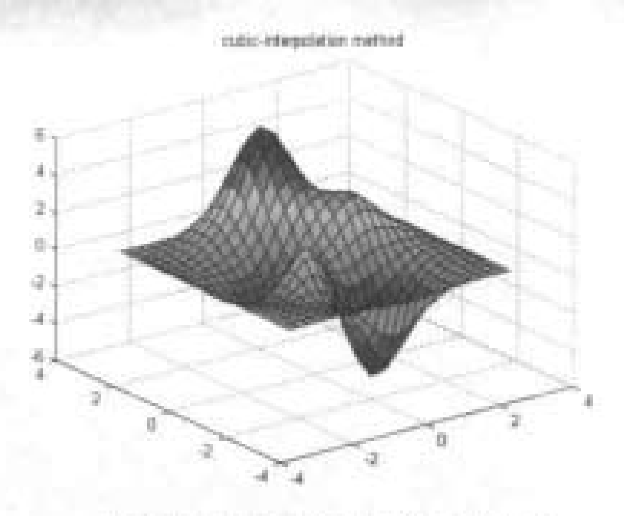



图 16-11 基于二维三次多项式插值数据的三维表面图（例 16-6）

可见，在已知数据点集比较疏散的情况下，用二维三次多项式插值可以获得最好的曲面光滑性，因此，这种插值方法在图像处理方面应用很广泛。

## 16.3 高维插值

MATLAB 中还提供了高维插值的函数：

（1）interp3

用于三维插值，语法格式为：

*VI*=interp3(*X*,*Y*,*Z*,*V*,*XI*,*YI*,*ZI*,method)

method 和二维插值类似，也有'nearest'，'linear'，'cubic'三种，其算法类似。

（2）interpn

用于 *n* 维插值，语法格式为：

*VI*=interpn(*X*1,*X*2,*X*3⋯,*V*,*Y*1,*Y*2,*Y*3,⋯,method)，同样有三种 method。

（3）ndgrid

用于产生 *n* 维空间上的栅格，高维插值中已知点栅格和插值点栅格都要用 ndgrid 函数产生。

ndgrid 的语法格式和产生二维/三维栅格点的 meshgrid 类似，是：

[*X*1,*X*2,*X*3,⋯]=ndgrid(*x*1,*x*2,*x*3,⋯)。

高维插值用法和二维插值类似，而且也没有一维插值和二维插值那么常用。

## 16.4 插值函数总结

表 16-5 总结了 MATLAB 中与插值相关的函数。

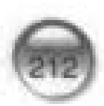


表 16-5　MATLAB 中插值相关函数

| 函　数 | 说　明 |
| --- | --- |
| interp1 | 一维插值函数 |
| spline/pchip | 一维样条插值/三次多项式插值 |
| ppval | 计算分段多项式在给定点的函数值 |
| interp2/interp3/interpn | 二维/三维/n 维插值函数 |
| meshgrid/ndgrid | 产生二（三）维/n 维空间的栅格点 |
| griddata | 由不等间距分布的点产生等间距分布的栅格点 |

**例 16-7**　griddata 在二维插值中的应用。

**解**：在命令窗口输入：

```
>> x = rand(100,1)*4-2;       %产生 100 个[-2 2]的均匀分布数据
>>  y = rand(100,1)*4-2;
>> z = x.*exp(-x.^2-y.^2);
>> ti = -2:.25:2;
>> [XI,YI] = meshgrid(ti,ti);   %产生精细的插值栅格点
>> ZI = griddata(x,y,z,XI,YI,'cubic');      %基于非栅格数据点和 cubic 方法估计
插值点函数值
>> mesh(XI,YI,ZI)
>> hold on        %打开附加绘图模式
>> plot3(x,y,z,'o')      %画出原始非栅格的离散数据点，如图 16-12 所示
>> hold off      %关闭附加绘图模式
```

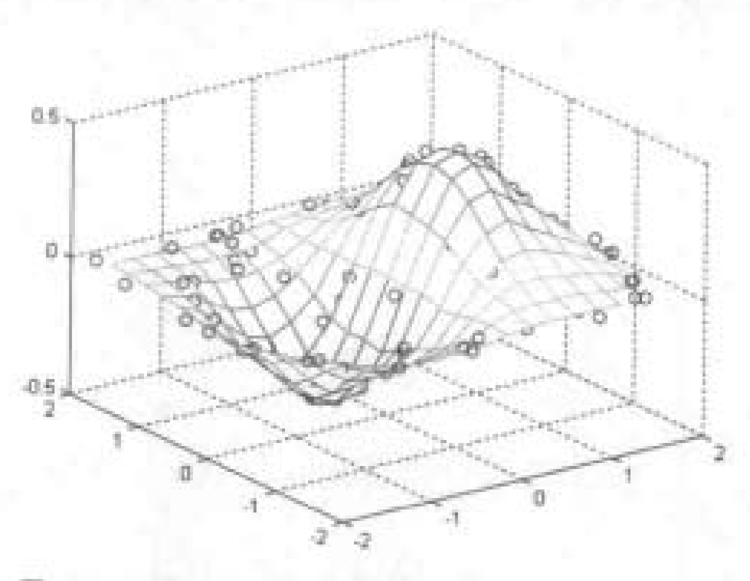



图 16-12　grdidata 对非栅格数据进行插值的结果（例 16-7）

## 16.5　小结

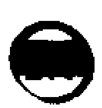
本章介绍 MATLAB 中插值相关的知识。插值是根据已知数据点集，预测点集内部其他点上函数值的方法，在信号处理和图像处理方面有重要和广泛的应用。



本章重点讲解了一维插值和二维插值的知识，比较了多种插值方法的优缺点，还通过图形可视化方法对插值效果进行了直观的比较。通过本章学习，读者应该熟练掌握一维和二维插值运算的各种方法，对于各种插值方法的原理和优劣也应该有明确的认识。

本章最后还简单讲述了 MATLAB 中高维插值的相关函数和通过非栅格点进行二维插值预测的 griddata 函数，这部分内容，读者简单了解就可以了。

另外，在数据插值部分，MATLAB 还提供了散列数据分析和几何分析的三角剖分方法，这部分内容本章没有提到，感兴趣的读者可以参阅 MATLAB 联机帮助。



# 第 17 章 多项式

MATLAB 提供了许多多项式处理的函数，这些函数在微分、积分等运算中有着广泛的应用，下面对其一一介绍。本章介绍 MATLAB 中多项式的表示、创建、运算和曲线拟合。

## 17.1 多项式基础

### 17.1.1 多项式的表示

在 MATLAB 中，用行向的一维数组表示多项式，数组元素为多项式的系数，并按照从高阶到低阶的顺序排列，比如多项式：

$$p(x)=x^3-4x^2+5x-2$$

在 MATLAB 中则用数组表示为：

**p**=[1 -4  5 -2]

这表示最高阶系数为 1，随着阶数递减，系数依次为-4，5，-2。由于共有 4 个系数，除去一个常数项，**p** 表示三次多项式。

当多项式中某些阶的系数为 0 时，在用数组表示此多项式时，必须用 0 补齐这些系数。比如多项式：

$$q(x)=x^5-3x^2+2$$

用数组表示则为：

**q**=[1 0 0 -3 0 2]

表示这个 5 次多项式的 4 次项、3 次项和 1 次项系数都为 0。

### 17.1.2 多项式的根

MATLAB 中可以用 roots 函数求解多项式的根，结果以列向的一维数组形式返回。

**例 17-1** 求解多项式 $s(x)=x^3-6x^2-72x-27$ 的根。

**解：** 在命令窗口输入：

```
>> p = [1 -6 -72 -27]
p =
     1    -6   -72   -27
>> r = roots(p)
r =
   12.1229
   -5.7345
   -0.3884
```

这对应于：

$$s(x)=x^3-6x^2-72x-27=(x-12.1229)(x+5.7345)(x+0.3884)$$

### 17.1.3 多项式的创建

创建多项式有两种方法：

（1）一种是直接输入系数数组；

（2）另一种是通过多项式的零点（根）用 poly 函数创建，poly 函数的输入参数是列向的一维数组。

**例 17-2** 多项式的创建。

**解：** 在命令窗口输入：

```
>> s=[1 5 -2 5]
s =
     1     5    -2     5
>> r=roots(s)
r =
  -5.5257
   0.2629 + 0.9142i
   0.2629 - 0.9142i
>> ss=poly(r)
ss =
    1.0000    5.0000   -2.0000    5.0000
```

可见，通过 roots 求根后再用 poly 可以得到多项式根对应的多项式的系数。有些情况下，只知道多项式的零点，则可以用 poly 函数创建多项式的系数数组。

poly 函数的输入参数还可以是二维数组，这时候 poly 函数返回的是该数组的特征多项式，该多项式的零点就是二维数组的特征值。



例 17-3 特征多项式。

解：在命令窗口输入：

```
>> A=rand(4)
A =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
>> p=poly(A)
p =
    1.0000   -2.7334    1.2135   -0.6543    0.1155
>> r=roots(p)
r =
   2.3230
   0.0914 + 0.4586i
   0.0914 - 0.4586i
   0.2275
>> e=eig(A)
e =
   2.3230
   0.0914 + 0.4586i
   0.0914 - 0.4586i
   0.2275
```

### 17.1.4 多项式求值

在工程计算中，经常需要计算给定多项式在某点的值。MATLAB 中可以用 polyval 函数来计算多项式在指定点的值。

例 17-4 多项式求值。

解：在命令窗口输入：

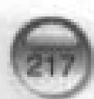
```
>> s=[3 5 -4 2]
s =
     3     5    -4     2
>> a=2.7
a =
    2.7000
>> polyval(s,a)
ans =
   86.6990
>> 3*a*a*a+5*a*a-4*a+2
ans =
   86.6990
```

polyvalm 可以接受二维数组形式的输入参数，对二维数组进行运算，这种运算要求输入数组是行列相等的方阵。

例 17-5 数组的多项式求值。

解：在命令窗口输入：

```
>> s=[3 5 -4 2]
s =
```



```
     3     5    -4     2
>> A=randn(4);
>> polyvalm(s,A)
ans =
   22.0075  -10.0686    8.4898  -25.5778
   -2.0601    4.0783    1.2236    1.5092
    1.2574    1.6111   21.1909   -5.4269
  -14.8213    3.3832  -22.7553   24.5802
>> 3*A*A*A+5*A*A-4*A+2*eye(size(A))
ans =
   22.0075  -10.0686    8.4898  -25.5778
   -2.0601    4.0783    1.2236    1.5092
    1.2574    1.6111   21.1909   -5.4269
  -14.8213    3.3832  -22.7553   24.5802
>> B=rand(2,4)
B =
    0.9355    0.4103    0.0579    0.8132
    0.9169    0.8936    0.3529    0.0099
>> polyvalm(s,B)
??? Error using ==> polyvalm
Matrix must be square.
```

## 17.2 多项式运算

### 17.2.1 多项式乘法

MATLAB 中提供了 conv 函数，可以进行多项式乘法运算。

需要注意的是：

（1）乘号（*）用于数组乘法，要求第一个数组的列数等于第二个数组的行数；

（2）点乘（.*）用于逐个元素的乘法，要求两个数组具有相同尺寸。

这些不要和多项式乘法相混淆。

**例 17-6** 多项式乘法。

**解**：在命令窗口输入：

```
>> a = [1 2 3];
>>  b = [4 5 6];
>> c = conv(a,b)
c =
     4    13    28    27    18
```

这表示：

$$(x^2+2x+3)*(4x^2+5x+6)=4x^4+13x^3+28x^2+27x+18$$

### 17.2.2 多项式除法

除法是乘法的逆运算，MATLAB 中多项式除法用 deconv 函数。

deconv 函数的完整语法是：



[*q*,*r*]=deconv(*v*,*u*)

其中 *q* 为商多项式，*r* 为余式多项式，该结果表示 *v*=conv(*q*,*u*)+*r*。对 deconv 只指定一个变量接收返回值时，则只接收 *q*。

**例 17-7** 多项式除法。

**解**：在命令窗口输入：

```
>> v=[3 5 2 1 4];
>> u=[2 5 3];
>> [q,r]=deconv(v,u)
q =
    1.5000   -1.2500    1.8750
r =
         0         0         0   -4.6250   -1.6250
>> conv(q,u)+r
ans =
     3     5     2     1     4
>> result=deconv(v,u)
result =
    1.5000   -1.2500    1.8750
```

这表示：

$$\frac{3x^4+5x^3+2x^2+x+4}{2x^2+5x+3}=(1.5x^2-1.25x+1.875)+\frac{-4.625x-1.625}{2x^2+5x+3}$$

### 17.2.3 多项式加法

多项式的加法其实就是相同阶数的系数相加的同类项合并运算。

（1）当两个多项式具有相同阶数时，它们的系数数组也有相同的尺寸，可以用加号（+）直接实现多项式的加法操作；

（2）当两个多项式阶数不同时，就要对阶数较低，也就是系数数组较短的那个左侧补零，直到两个系数数组尺寸相同时，再用加号进行加法操作。这个可以编写函数实现。

**例 17-8** 多项式加法。

**解**：在命令窗口输入

编写的多项式加法函数 polyplus：

```
%Ex1708    polynomials addition
function y=polyplus(a,b)
%evaluate the plus of two polynomials
%a and b are the input arguments, in form of vectors.
%return a vector representing the plus result.
%creat on 2006/01/20
la=length(a);lb=length(b);
if la==lb
   y=a+b;
elseif la>lb
   y=a+[zeros(1,la-lb) b];
else
   y=[zeros(1,lb-la) a]+b;
```

```
end
```

命令窗口运行代码：

```
>> a=rand(1,4)
a =
    0.6813    0.3795    0.8318    0.5028
>> b=randn(1,4)
b =
    0.8156    0.7119    1.2902    0.6686
>> c=rand(1,3)
c =
    0.7095    0.4289    0.3046
>> a+b
ans =
    1.4969    1.0914    2.1220    1.1714
>> polyplus(a,b)
ans =
    1.4969    1.0914    2.1220    1.1714
>> a+c
??? Error using ==> plus
Matrix dimensions must agree.

>> a+[0 c]
ans =
    0.6813    1.0890    1.2607    0.8074
>> polyplus(a,c)
ans =
    0.6813    1.0890    1.2607    0.8074
```

### 17.2.4 多项式微分

MATLAB 中对多项式进行微分操作要用到 polyder 函数。为了处理多种微分情况，polyder 有多种语法格式：

（1）***k***=polyder(***p***)直接计算 ***p*** 的微分多项式 ***k***，表示 $k = p'$；

（2）***k***=polyder(***a***,***b***)计算 conv(***a***,***b***)的微分多项式 ***k***，表示 $k = (conv(a,b))'$；

（3）[***q***,***d***]=polyder(***a***,***b***)计算分式 ***a***/***b*** 的微分结果，相当于 $q/d = (a/b)'$。

**例 17-9** 多项式微分。

**解**：在命令窗口输入：

```
>> a=[3,5,7,2]
a =
     3     5     7     2
>> b=[4,3,1]
b =
     4     3     1
>> polyder(a)
ans =
     9    10     7
>> polyder(a,b)
ans =
    60   116   138    68    13
```

```
>> [q,d]=polyder(a,b)
q =
    12    18    -4    -6     1
d =
    16    24    17     6     1
```

结果的意义分别是：

$$(3x^3+5x^2+7x+2)'=9x^2+10x+7$$

$$((3x^3+5x^2+7x+2)*(4x^2+3x+1))'=60x^4+116x^3+138x^2+68x+13$$

$$(\frac{3x^3+5x^2+7x+2}{4x^2+3x+1})'=\frac{12x^4+18x^3-4x^2-6x+1}{16x^4+24x^3+17x^2+6x+1}$$

### 17.2.5 多项式的部分分式展开

MATLAB 提供了 residue 函数，可以将两个多项式相除用部分分式展开形式来表示。

residue 的完全调用的语法格式为：

**[*r*,*p*,*k*]**=residue(***a***,***b***)

结果表示 ***a*/*b*** 的商多项式为 ***k***，余式多项式可以表示成多个分式的和。

这些分式的分子依次为 ***r*** 的每一个元素，分母为一次多项式，其零点依次为 ***p*** 的每一个元素，即：

$$\frac{a(x)}{b(x)}=k(x)+\frac{r(1)}{x-p(1)}+\frac{r(2)}{x-p(2)}+\cdots+\frac{r(n)}{x-p(n)}$$

residue 还可以把部分分式和的形式转化成两个多项式除法的形式，其格式为：

**[*a*,*b*]**=residue(***r***,***p***,***k***)

意义与上述类似，即：

$$k(x)+\frac{r(1)}{x-p(1)}+\frac{r(2)}{x-p(2)}+\cdots+\frac{r(n)}{x-p(n)}=\frac{a(x)}{b(x)}$$

**例 17-10** 多项式的部分分式展开。

**解**：在命令窗口输入：

```
>> a=[1 3 5 7];
>> b=[1 5 6];
>> [r,p,k]=residue(a,b)
r =
    8.0000
    1.0000
p =
   -3.0000
   -2.0000
k =
     1    -2
>> [aa,bb]=residue(r,p,k)
aa =
    1.0000    3.0000    5.0000    7.0000
bb =
```

```
        1     5     6
```

此计算结果可以表示为：

$$\frac{x^3+3x^2+5x+7}{x^2+5x+6}=(x-2)+\frac{8}{x-(-3)}+\frac{1}{x-(-2)}$$

## 17.3 多项式曲线拟合

曲线拟合是数据分析中常用的方法，即是在两组数据之间建立一种已知形式的函数关系，使得通过这种函数关系预测得到的数据结果和实际测量的数据最大程度的吻合，这在工程应用和科学研究中都有很广泛的应用。当待拟合的函数关系是多项式形式的函数，则称为多项式曲线拟合。

MATLAB 中多项式曲线拟合的函数是 polyfit，其语法格式为：

***p*=polyfit(*x*,*y*,*n*)**

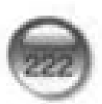
它返回一个 *n* 阶多项式的系数数组 ***p***，表示 polyval((p, x(i))能在最小二乘意义上拟合 *y*(*i*)。

**例 17-11** 多项式的曲线拟合。

**解**：在命令窗口输入：

```
>> x=0:0.5:20;
>> y=polyval([3,5,1,2],x)+randn(size(x));          %计算已知函数并加入随机误差
>> p1=polyfit(x,y,1)
p1 =
  1.0e+003 *
    1.1898   -5.0616
>> y1=polyval(p1,x);
>> p2=polyfit(x,y,2)
p2 =
  1.0e+003 *
    0.0950   -0.7101    1.1133
>> y2=polyval(p2,x);
>> p3=polyfit(x,y,3)
p3 =
    2.9998    5.0047    0.9636    1.9170
>> y3=polyval(p3,x);
>> plot(x,y,'.',x,y1,'-.',x,y2,'--',x,y3,'-')   %标出带有误差的数据点，用不同线型画出拟合曲线
```

例 17-11 对一个已知形式的三次多项式的数据点增加随机误差，然后对有误差的数据进行不同阶次的多项式曲线拟合，最后用 plot 函数画出这些数据点和拟合曲线。从拟合曲线（图 17-1）可以看出，二次多项式的拟合效果已经比较好，三次多项式的拟合结果几乎和加入误差前函数原本形式一致。



原书缺页

```
pp =
    0.9734   -0.1117   -4.8399    0.7623    9.3169
>> ypp=polyval(pp,x);
>> plot(x,y,x,ys,'o',x,ypp,'-.')
>> grid on        %打开图像显示中的栅格，结果如图 17-2 所示
```

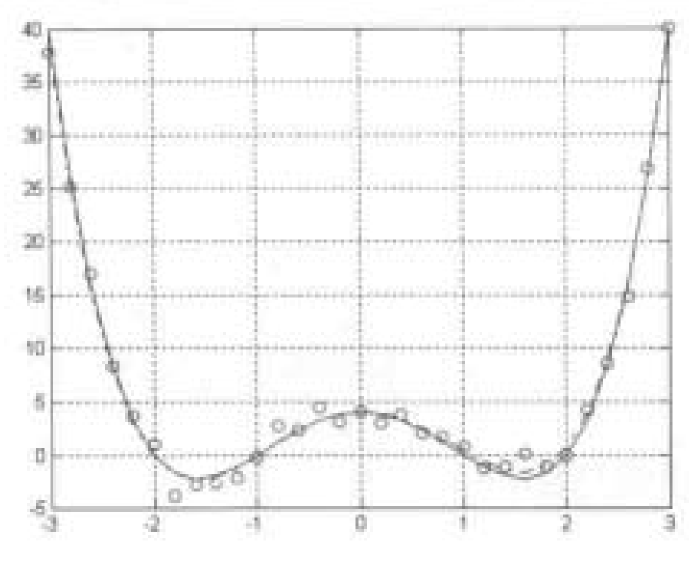



图 17-2　例 17-12 曲线拟合结果

从图 17-2 结果可以看出，蓝色连续线条表示的函数确实具有 4 个零点，分别是–2，–1，1，2。加入随机误差后，数据点偏离标准函数曲线，如圆圈所示。

经过对带误差数据进行 4 次多项式的曲线拟合后，得到的多项式结果和原标准多项式非常接近，这是因为加入的随机误差没有改变原数据的基本分布特性。

## 17.5　小结

本章在讲述了 MATLAB 中多项式的数组表示方法的基础上，介绍了众多多项式的操作方法，包括：创建、求根、加法、乘除、微分，以及多项式求值和多项式拟合等知识。其中，多项式拟合是应用中的重点，但要用好多项式拟合又必须以本章讲到的其他操作为基础，如本章例 17-11 多项式曲线拟合中，就用到了多项式求值。

通过本章的学习，读者要熟练掌握 MATLAB 中操作多项式的各种方法，能够熟练应用多项式的曲线拟合。

# 第 18 章 三次样条

由于高阶多项式插值通常产生病态结果，三次样条是消除病态结果的最常用方法。MATLAB 提供了实现三次样条的函数，易于使用，下面详细介绍这些函数。

## 18.1 三次样条基础

为了用更光滑的曲线来拟合数据点，最常用的方法是用一个三阶多项式，即三次多项式，来对相邻数据点之间的各段建模，每个三次多项式的头两个导数在该数据点相一致。这种类型的插值称为三次样条，或简称为样条。

样条函数是由一些按照某种光滑条件分段拼接起来的多项式组成的函数。高阶多项式插值常常产生病态的结果，而三次样条是最常用的一种消除病态的方法。在三次样条中，要寻找三次多项式，以逼近每对数据点间的曲线，这些数据点称为断点。由于两点只能决定一条直线，而在两点间的曲线可用无限多的三次多项式近似，因此为使结果具有唯一性，在三次样条中，增加了三次多项式的约束条件。通过限定每个三次多项式的一阶和二阶导数，使其在断点处相等，就可以较好地确定所有内部三次多项式。此外，近似多项式通过这些断点的斜率和曲率是连续的。

## 18.2 三次样条的 MATLAB 实现

在 MATLAB 中，实现基本的三次样条插值的函数有 spline，其语法格式如下：

（1）***yy*** = spline(***x***,***y***,***xx***)

（2）***pp*** = spline(***x***,***y***)

对于语法格式：

*yy* = spline(*x*,*y*,*xx*)

其中：**x** 为节点（*n* 维向量），**y** 为函数值（可以是 $m \times n$ 维矩阵），**xx** 为插值点（可以是向量）。

对于给定的离散测量数据 *x*,*y*（称为断点），要寻找一个三项多项式 $y=p(x)$，以逼近每对数据（*x*,*y*）点间的曲线。过两点 $(x_i, y_i)$ 和 $(x_{i+1}, y_{i+1})$ 只能确定一条直线，而通过一点的三次多项式曲线有无穷多条。为使通过中间断点的三次多项式曲线具有唯一性，要增加两个条件（因为三次多项式有 4 个系数）：

（1）三次多项式在点 $(x_i, y_i)$ 处有：$p_i(x_i) = p_i^{*}(x_i)$；

（2）三次多项式在点 $(x_{i+1}, y_{i+1})$ 处有：$p_i(x_{i+1}) = p_i^{*}(x_{i+1})$；

（3）$p(x)$在点 $(x_i, y_i)$ 处的斜率是连续的（为了使三次多项式具有良好的解析性而加上的条件）；

（4）$p(x)$在点 $(x_i, y_i)$ 处的曲率是连续的。

对于第一个和最后一个多项式，人为地规定如下条件：$p_1'''(x) = p_2'''(x)$ 和 $p_n'''(x) = p_{n-1}'''(x)$ 上述两个条件称为非结点(not-a-knot)条件。

由上可知，对数据拟合的三次样条函数 $p(x)$是一个分段的三次多项式：

$$p(x)=\begin{cases} p_1(x) & x_1 \leqslant x \leqslant x_2 \\ p_2(x) & x_2 \leqslant x \leqslant x_3 \\ \cdots & \cdots \\ p_n(x) & x_n \leqslant x \leqslant x_{n+1} \end{cases}$$

其中每段 $p_i(x)$ 都是三次多项式。

**spline** 命令用三次样条插值计算出由向量 **x** 与 **y** 确定的一元函数 $y=f(x)$在点 **xx** 处的值。若参量 **y** 是一矩阵，则以 **y** 的每一列和 **x** 配对，再分别计算由它们确定的函数在点 **xx** 处的值。则 **yy** 是一阶数为 **length(xx)*size(y,2)**的矩阵。

对于语法格式：

*pp* = spline(*x*,*y*)

其中：返回一个结构类型的数据 *pp*，需用 unmkpp 函数解开，unmkpp 函数的调用格式如下：

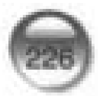
[breaks,coefs,nploys,ncofs,dim] = unmkpp(*pp*)

其中，breaks 是节点，coefs 是个矩阵，nploys 是多项式个数 *n*，ncofs 是每个多项式的系数个数，dim 是维数 *m*。

**例 18-1** 产生一条正弦曲线，然后用三次样条插值进行拟合，并对比拟合结果和原数据。

**解**：在命令窗口输入：

```
>> x = 0:10
y = sin(x);
xx = 0:.25:10;
```





```
yy = spline(x,y,xx);  %三次样条插值
plot(x,y,'o',xx,yy)   %将实际曲线与样条断点在同一图上显示
%输出为
x =
  Columns 1 through 9
     0     1     2     3     4     5     6     7     8
  Columns 10 through 11
     9    10
```

程序运行后，输出如图 18-1 所示的曲线。

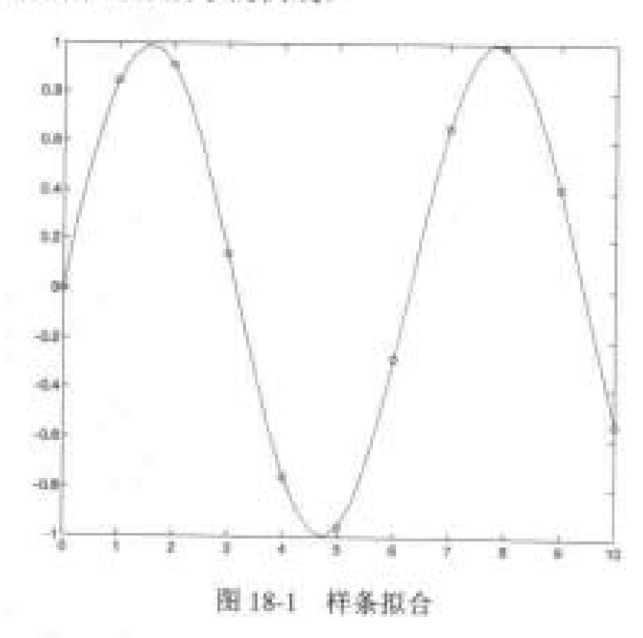


图 18-1　样条拟合

下面计算三次样条的 *pp* 形式，在命令窗口中输入：

```
>> pp = spline(x,y);
>> pp
%输出为
pp =
      form: 'pp'
    breaks: [0 1 2 3 4 5 6 7 8 9 10]
     coefs: [10x4 double]
    pieces: 10
     order: 4
       dim: 1
```

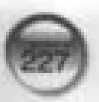
此处的三次样条 *pp* 形式，采用结构变量的形式保存了断点和多项式系数，以及关于三次样条的其他信息，其中 ***breaks*** 表示断点的位置；***coefs*** 表示多项式系数矩阵，它的 *i* 行是第 *i* 个多项式的系数；***pieces*** 表示多项式的数目；***order*** 表示每个多项式的阶数；***dim*** 表示插值的维数。

得到 *pp* 形式后，可以采用函数 ppval 计算该三次样条。该函数的用法如下：

*yi*=ppval(*pp*,*xi*)

对于相同的 *pp* 形式，指定不同的 *xi* 可以得到不同的三次样条插值。

当要计算三次样条表示时，必须把 *pp* 形式分解成它的各个部分。在 MATLAB 中，通



过函数 unmkpp 完成这一功能。

反之，如果给定 *pp* 形式的各个部分，可以用函数 mkpp 重构完整的 *pp* 形式。

继续上面的例子，进行分解和重构 *pp* 形式，在命令窗口中输入：

```
>> [breaks,coefs,nploys,ncofs,dim] = unmkpp(pp);    %分解 pp 形式
%输出为
breaks =
     0     1     2     3     4     5     6     7     8     9    10
coefs =
   -0.0419   -0.2612    1.1446         0
   -0.0419   -0.3868    0.4965    0.8415
    0.1469   -0.5124   -0.4027    0.9093
    0.1603   -0.0716   -0.9867    0.1411
    0.0372    0.4095   -0.6488   -0.7568
   -0.1234    0.5211    0.2818   -0.9589
   -0.1684    0.1509    0.9538   -0.2794
   -0.0640   -0.3542    0.7506    0.6570
    0.1190   -0.5463   -0.1499    0.9894
    0.1190   -0.1894   -0.8856    0.4121
nploys =
    10
ncofs =
     4
dim =
     1
```

在命令窗口中输入：

```
>> pp1 = mkpp(breaks, coefs);    %重构 pp 形式
>> pp1
%输出为
pp1 =
      form: 'pp'
    breaks: [0 1 2 3 4 5 6 7 8 9 10]
     coefs: [10x4 double]
    pieces: 10
     order: 4
       dim: 1
```

从上面可以看出，只需要断点位置向量 ***breaks*** 和多项式系数矩阵 ***coef*** 两个参数就可以重构出完全相同的 *pp* 形式，这是因为矩阵 ***coef*** 的大小确定了参数 *pieces* 和 *order*，所以 mkpp 不需要 *pieces* 和 *order* 去重构 *pp* 形式。

## 18.3 小结

本章讲述了三次样条的基本概念，介绍了 MATLAB 中三次样条函数的使用，并通过实例对这些函数进行了介绍。



# 第 19 章

# 傅里叶分析

傅里叶级数、傅里叶变换以及它们在离散时间域对应变换构成了信号处理的基础，MATLAB 提供了相应的函数进行傅里叶分析，完成许多信号处理任务。

## 19.1 傅里叶变换

信号频域分析是采用傅里叶变换将时域信号 $x(t)$变换为频域信号 $X(f)$，从而帮助人们从另一个角度来了解信号的特征，傅里叶变换的公式为：

$$F(jw)=\int_{-\infty}^{+\infty}f(t)e^{-jwt}dt$$

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}F(w)e^{jwt}dw$$

在 MATLAB 中，符号数学工具箱提供了一些函数，能够进行解析的傅里叶变换及逆变换，下面介绍这些函数的用法。

### 1. 傅里叶变换

（1）*F*=fourier(*f*)：符号表达式 *f* 的傅里叶变换。默认的自变量为 *x*，默认返回值是关于 w 的函数。如果 *f*=*f*(w)，fourier 函数返回关于 *t* 的函数。

（2）*F*=fourier(*f*,*v*)：返回函数 *F* 是关于符号表达式对象 *v* 的函数，而不是默认的 *w*。

（3）*F*=fourier(*f*,*u*,*v*)：对关于 *u* 的函数 *f* 进行变换，返回函数 *F* 是关于 *v* 的函数。

### 2. 傅里叶逆变换

（1）*f*=ifourier(*F*)：符号表达式对象的傅里叶逆变换。默认的自变量为 *w*，默认返回 *x* 的函数。如果 *F*=*F*(*x*)，ifourier 返回关于 *t* 的函数。

（2）*f*=ifourier(*F*,*u*)：返回函数 *f* 是关于符号表达式对象 *u* 的函数，而不是默认的 *x* 的函数。

（3）*f*=ifourier(*F*,*v*,*u*)：对关于 *v* 的函数 *F* 进行变换，返回关于 *u* 的函数 *f*。

**例 19-1** 求函数 $f(x)=e^{-x^2}$ 的傅里叶变换及其逆变换。

**解：** 在命令窗口输入：

```
>> syms x;
f = exp(-x^2)
f1=fourier(f)     %求 f 的傅里叶变换
f2=ifourier(f1)   %求 f1 的傅里叶逆变换
%输出为
f =
exp(-x^2)
f1 =
pi^(1/2)*exp(-1/4*w^2)
f2 =
exp(-x^2)
```

**例 19-2** 求函数 $y(t)=|t|$ 的傅里叶变换及其逆变换。

**解：** 在命令窗口输入：

```
>> syms w t;
y=abs(t);
F=fourier(y,t,w)      %求 y 的傅里叶变换
f=ifourier(F,w,t)     %求 F 的傅里叶逆变换
%输出为
F =
-2/w^2
f =
t*(2*heaviside(t)-1)
```

其中，Heaviside 函数为单位阶跃函数。

## 19.2 快速傅里叶变换（FFT）

傅里叶变换把信号按正弦展开成不同的频率值，对于取样信号，用的是离散傅里叶变换。一般信号波形的记录都是以时间-幅度相关的形式直观表现出来的，称为时域分析；而快速傅里叶变换就是分析计算信号波形中的频谱成分强度，将其能量从时间积分，从而得出频率-能量相关的形式，称为频域分析。频率-能量相关并非一般认为的频率-幅度相关，因为傅里叶变换实际上已经无法确认信号不同频率间的幅度关系，而只能计算出其能量关系。

对一个时域离散信号 $\{x(n)\}$，其频谱函数 $X(j\omega)$ 是 $x(n)$ 的傅里叶变换。傅里叶变换定义为：

$$X(j\omega)=\sum_{n=-\infty}^{\infty}x(n)e^{-j\omega n}$$

其中 j 为虚数单位，$\omega$ 称为数字频率。

由于处理的信号是有限长的，即 $n<\infty$，故实际采用的是离散傅里叶变换 DFT (Discrete Fourier Transform)。

长度为 $N$ 的序列 $x(n)$，其 DFT 定义为：

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j(2\pi/N)kn}=\sum_{n=0}^{N-1}x(n)W_N^{kn}$$

其中，$W_N=e^{-j(2\pi/N)}$。

如果 $x(n)$为一个周期序列，得到的 $X(k)$为 $x(n)$的周期频谱。若 $x(n)$不是周期序列，先对 $x(n)$进行周期性扩展，把它看成某个周期性信号的一个周期，得到的 $X(k)$是 $x(n)$频谱在一个周期的采样。

DFT 有一种快速算法 FFT，称为快速傅里叶变换。FFT 并不是与 DFT 不同的另一种变换，而是为了减少 DFT 运算次数的一种快速算法。它是对变换式进行一次次分解，使其成为若干小数点的组合，从而减少运算量。常用的 FFT 是以 2 为基数的，其长度用 $N$ 表示，$N$ 为 2 的整数倍。

从频率采样定理知道，$N$ 点序列 $x(n)$的 $N$ 个离散时间傅里叶变换 $X(e^{j\omega})$等间隔样本能惟一地重构 $x(n)$。这些单位圆上的 $N$ 个样本叫做离散傅里叶系数。设 $\widetilde{X}(k)=DFS\,\widetilde{x}(n)$ 为一周期（具有无限持续时间）序列，则它的主周期为具有有限持续时间的离散傅里叶变换，$N$ 点序列的离散傅里叶变换由下式给出：

$$X(k)\overset{\Delta}{=}DFT\left[x(n)\right]=\begin{cases}\widetilde{X}(k) & 0\leqslant k\leqslant N-1\\ 0 & \text{其他}\end{cases}=\widetilde{X}(k)R_N(k)$$

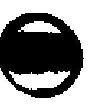
N 点序列的离散傅里叶反变换为：

$$x(n)\overset{\Delta}{=}IDFT\left[X(k)\right]=\tilde{x}(n)R_N(n)$$

MATLAB 中采用的就是 FFT 算法。在 MATLAB 中，函数 fft 计算一个信号的离散傅里叶变换。在数据的长度是 2 的幂次或质因数的乘积的情况下，就用快速傅里叶变换来计算离散傅里叶变换。当数据长度是 2 的幂次时，计算速度显著增加，因此，只要可能，选择数据长度为 2 的幂次或者用零来填补数据，使得数据长度等于 2 的幂次显得非常重要。MATLAB 中提供的进行信号处理的函数如表 19-1 所示。

**表 19-1　MATLAB 中的信号处理函数**

| 函数名称 | 描　述 |
|---|---|
| fft | 离散傅里叶变换 |
| fft2 | 二维离散傅里叶变换 |
| fftn | $n$ 维离散傅里叶变换 |
| ifft | 离散傅里叶反变换 |
| ifft2 | 二维离散傅里叶反变换 |
| ifftn | $n$ 维离散傅里叶反变换 |
| abs | 幅值 |
| angle | 4 个象限的相角 |



续表

| 函数名称 | 描　　述 |
|---|---|
| unwrap | 相位按弧度展开，大于 π 的变换为 2π 的补角 |
| fftshift | 把 FFT 结果平移至负频率上 |
| cplxpair | 把数据排成复数对 |
| nextpow2 | 下两个更高的功率 |

下面介绍 MATLAB 中这些用于快速傅里叶变换的函数用法。

### 1．fft 和 ifft

函数 fft 和 ifft 对数据作一维快速傅里叶变换和反傅里叶变换，函数 fft 的调用格式有如下几种：

（1）*Y*=fft(*X*)　如果 *X* 是向量，则采用快速傅里叶变换算法作 *X* 的离散傅里叶变换；如果是矩阵，则计算矩阵每一列的傅里叶变换；如果是多维数组，则对第一个非单元素的维进行计算。

（2）*Y*=fft(*X*, *n*)　用参数 *n* 限制 *X* 的长度，如果 *X* 的长度小于 *n*，则用 0 补足，如果 *X* 的长度大于 *n*，则去掉长出的部分。

（3）*Y*=fft(*X*, [], *n*)或 *Y*=fft(*X*, *n*, dim)　在参数 dim 指定的维上进行操作。

函数 ifft 的用法和 fft 完全相同。

### 2．fft2 和 ifft2

函数 fft2 和 ifft2 对数据作二维快速傅里叶变换和反傅里叶变换，数据的二维傅里叶变换 fft2(*X*)相当于 fft(fft(*X*)')'，即先对 *X* 的列做一维傅里叶变换，然后对变换结果的行做一维傅里叶变换。函数 fft2 的调用格式有如下几种。

（1）*Y*=fft2(*X*)：二维快速傅里叶变换。

（2）*Y*=fft2(*X*, MROWS, NCOLS)：通过截断或用 0 补足，使得 *X* 成为 MROWS* NCOLS 的矩阵。

函数 iff2 的用法和 fft2 完全相同。

和 fft2，ifft2 类似，fftn，ifftn 对数据作多维快速傅里叶变换，相关内容请参看 MATLAB 帮助。

### 3．fftshift 和 ifftshift

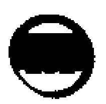
函数 fftshift(*Y*)用于把傅里叶变换结果 *Y*（频域数据）中的直流分量（频率为 0 处的值）移到中间位置：

（1）如果 *Y* 是向量，则交换 *Y* 的左右半边；

（2）如果 *Y* 是矩阵，则交换其一三象限和二四象限；

（3）如果 *Y* 是多维数组，则在数组的每一维交换其“半空间”。

函数 ifftshift 相当于把 fftshift 函数的操作逆转，用法相同。





**例 19-3**　傅里叶变换通常用于分析受噪声干扰的时域信号中的频率成分。以下是对受零均值的随机噪声干扰，包含 50Hz 频率和 120Hz 频率的信号进行分析。

**解：** 在命令窗口输入：

```
>> t = 0:0.001:0.6;                          %采样频率为 1000Hz
x = sin(2*pi*50*t)+sin(2*pi*120*t);          %产生正弦波
y = x + 2*randn(size(t));                    %叠加随机噪声
plot(1000*t(1:50),y(1:50))                   %绘制 y 的曲线
title('信号受零均值的随机噪声干扰')
xlabel('时间(毫秒)')
```

程序运行后，输出如图 19-1 所示的图形。

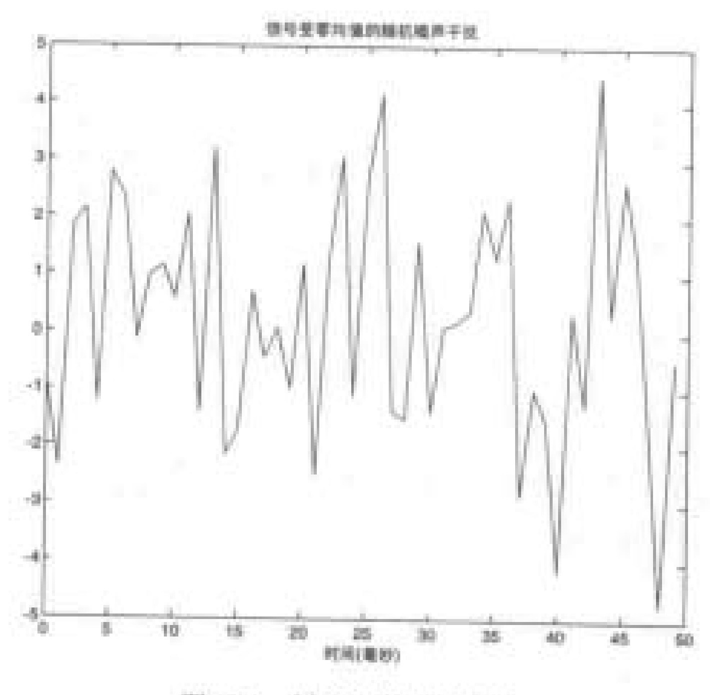



图 19-1　例 19-3 的受扰信号

19-1 中显示的是时域信号，信号的时域分析只能反映信号的幅值随时间的变化情况，除单频率分量的简谐波外，很难明确揭示信号的频率组成和各频率分量大小。

把时域信号变换到频域，采用离散傅里叶变换对受扰信号进行处理，进行 512 点的快速傅里叶变换，程序代码如下：

```
>>Y = fft(y,512);            %对 y 作傅里叶变换，取 512 个点
Pyy = Y.* conj(Y) / 512;
f = 1000*(0:256)/512;        %设置频率轴（横轴）坐标，1000 是采样频率
plot(f,Pyy(1:257))           %绘制频域图
title('y 的频率成分')
xlabel('频率 (Hz)')
```

程序运行后，输出如图 19-2 所示的图形。

从图 19-2 中可以明显看出，信号包含 50Hz 和 120Hz 频率的成分。



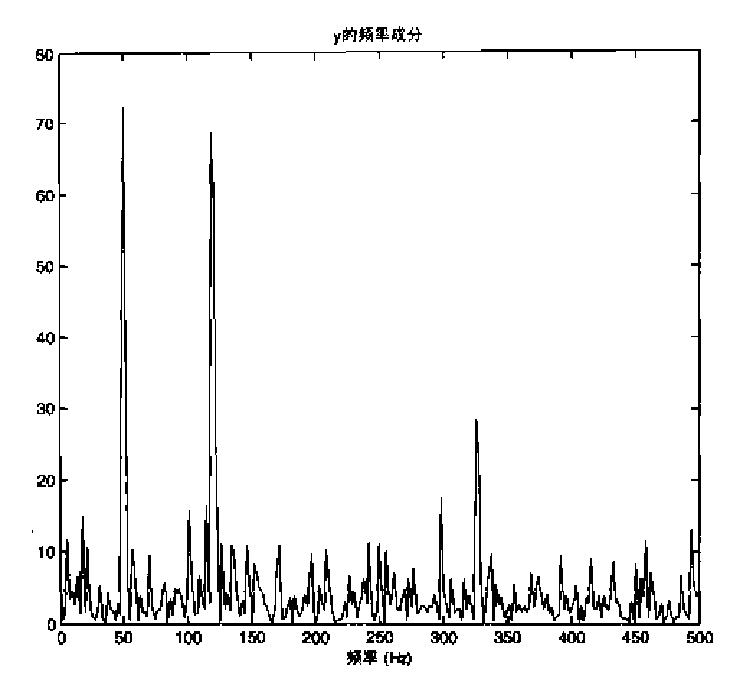
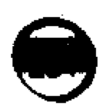



图 19-2　例 19-3 进行 FFT 之后的图形

## 19.3　小结

本章对 MATLAB 中进行傅里叶变换、快速傅里叶变换的方法进行了简要介绍，重点介绍了实现这些变换的 MATLAB 函数。



# 第 20 章 最优化计算

最优化计算在实际中有着广泛的应用，MATLAB 提供了进行最优化计算的工具箱，用户可以方便地进行最优化计算。

## 20.1 优化工具箱简介

### 20.1.1 优化工具箱 3.0 的新特色

MATLAB 7.x 提供的优化工具箱是最新的 3.0 版本，3.0 版本较之以前的 2.3 版本，主要增加了以下新的特色：

（1）二进制整数编程；

（2）用于 fminunc 函数的新的中规模算法；

（3）输出 hessian 返回有限差分逼近；

（4）exitflag 的新值；

（5）输出参数的新字段；

（6）FunValCheck 的新选项；

（7）优化函数调用提供参数的新方式。

优化工具箱主要可以用于解决以下问题：

（1）求解无约束条件非线性极小值；

（2）求解约束条件下非线性极小值，包括目标逼近问题、极大-极小值问题以及半无限极小值问题；

（3）求解二次规划和线性规划问题；

（4）非线性最小二乘逼近和曲线拟合；



（5）非线性系统的方程求解；
（6）约束条件下的线性最小二乘优化；
（7）求解复杂结构的大规模优化问题。

## 20.1.2 优化函数

3.0 版的优化工具箱的结构图如图 20-1 所示。

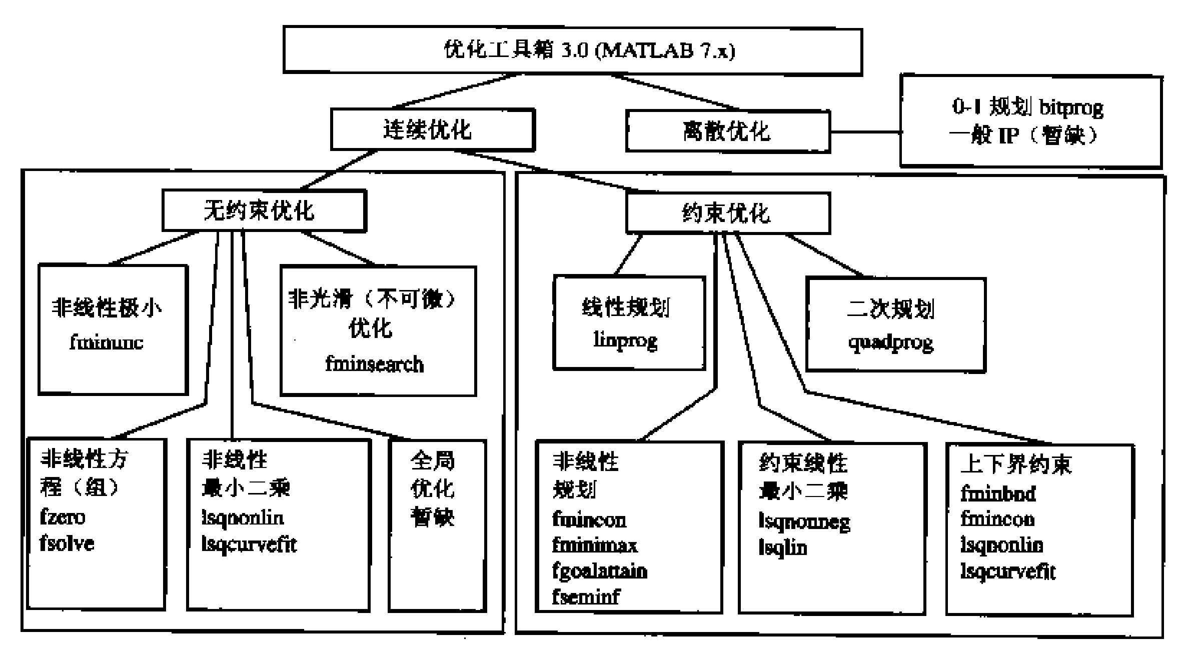
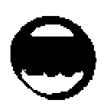



图 20-1 优化工具箱结构图

MATLAB 求解优化问题的主要函数如表 20-1 所示。

表 20-1 MATLAB 求解优化问题的主要函数

| 类　型 | 模　型 | 基本函数 |
|---|---|---|
| 一元函数极小 | Min $F(x)$<br>s.t.$x1<x<x2$ | $x$=fminbnd('F',$x_1$,$x_2$) |
| 无约束极小 | Min $F(X)$ | $X$=fminunc('F',$X_0$)<br>$X$=fminsearch('F',$X_0$) |
| 线性规划 | Min $c^T X$<br>s.t.$AX \leqslant b$ | $X$=linprog($c$,$A$,$b$) |
| 二次规划 | Min $\frac{1}{2}x^T Hx + c^T x$<br>s.t. $Ax \leqslant b$ | $X$=quadprog($H$,$c$,$A$,$b$) |
| 约束极小<br>（非线性规划） | Min $F(X)$<br>s.t. $G(X) \leqslant 0$ | $X$=fmincon('FG',$X_0$) |
| 达到目标问题 | Min $r$<br>s.t. $F(x)-wr \leqslant goal$ | $X$=fgoalattain('F',$x$,$goal$,$w$) |



续表

| 类　型 | 模　型 | 基本函数 |
|---|---|---|
| 极小极大问题 | $\min\limits_{X} \max\limits_{\{F_i(x)\}} \{F_i(x)\}$<br>s.t. $G(x) \leqslant 0$ | X=fminimax('FG',$x_0$) |

使用优化函数或优化工具箱中其他优化函数时，输入变量如表 20-2 所示。

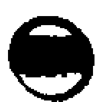
**表 20-2　优化函数的输入变量**

| 变　量 | 描　述 | 调用函数 |
|---|---|---|
| *f* | 线性规划的目标函数 *f**X 或二次规划的目标函数 X'*H*X+*f**X 中线性项的系数向量 | linprog,quadprog |
| *fun* | 非线性优化的目标函数.fun 必须为行命令对象或 M 文件、嵌入函数、或 MEX 文件的名称 | fminbnd,fminsearch,fminunc,<br>fmincon,lsqcurvefit,lsqnonlin,<br>fgoalattain,fminimax |
| *H* | 二次规划的目标函数 X'*H*X+*f**X 中二次项的系数矩阵 | quadprog |
| *A,b* | *A* 矩阵和 *b* 向量分别为线性不等式约束：$AX \leqslant b$ 中的系数矩阵和右端向量 | linprog,quadprog,fgoalattain, fmincon, fminimax |
| *Aeq*,<br>*beq* | *Aeq* 矩阵和 *beq* 向量分别为线性等式约束：$Aeq \cdot X = beq$ 中的系数矩阵和右端向量 | linprog,quadprog,fgoalattain, fmincon, fminimax |
| *vlb*,<br>*vub* | X 的下限和上限向量：$vlb \leqslant X \leqslant vub$ | linprog,quadprog,fgoalattain,<br>fmincon,fminimax,lsqcurvefit,<br>lsqnonlin |
| *X0* | 迭代初始点坐标 | 除 fminbnd 外所有优化函数 |
| *x1,x2* | 函数最小化的区间 | fminbnd |
| *options* | 优化选项参数结构，定义用于优化函数的参数 | 所有优化函数 |

优化函数时的输出变量如表 20-3 所示。

**表 20-3　优化函数的输出变量**

| 变　量 | 描　述 | 调用函数 |
|---|---|---|
| *x* | 由优化函数求得的值。若 *exitflag*>0，则 *x* 为解；否则，*x* 不是最终解，它只是迭代终止时优化过程的值 | 所有优化函数 |
| *fval* | 解 *x* 处的目标函数值 | linprog,quadprog,fgoalattain,<br>fmincon,fminimax,lsqcurvefit,<br>lsqnonlin, fminbnd |
| *exitflag* | 描述退出条件：<br>1．*exitflag*>0，表目标函数收敛于解 *x* 处<br>2．*exitflag*=0，表已达到函数评价或迭代的最大次数<br>3．*exitflag*<0，表目标函数不收敛 | |
| *output* | 包含优化结果信息的输出结构：<br>1．Iterations：迭代次数<br>2．Algorithm：所采用的算法<br>3．FuncCount：函数评价次数 | 所有优化函数 |



## 20.2 无约束优化问题

多数的最优化方法的基本思想都是由迭代算法而来，无约束最优化方法的主要步骤为：

（1）选定初始点 $x_0$，计算目标函数初始值 $f$（$x_0$）；

（2）选取一个能使目标函数值下降的方向，沿该方向取一下降点 $x_1$，能使目标函数值下降，即 $f$（$x_1$）$<f$（$x_0$）；

（3）当不存在下降方向，或虽存在但 $x_1$ 点与 $x_0$ 点已足够靠近，则认为找到了一个最优解，结束求解过程；否则，$x_0=x_1$，转向步骤（2）继续。

常用的无约束最优化方法有 Powell 法、梯度法、共轭梯度法、牛顿法、DFP 法（Davidon-Fletcher-Powell 法）等。不同方法之间的差别主要是用不同的方法选取下降方向和下降点。

### 20.2.1 一元函数无约束优化

许多方法中均包含沿下降方向找下降点的问题，这就构成了一个一维搜索问题。求解一维搜索问题的最优化方法有黄金分割法、二次插值法等。也就是说，无约束最优化方法的求解，是通过将求解一个多维最优化问题转化为求解一系列的一维搜索问题来实现的。

在 MATLAB 中，对形如 $\min f(x)$，$x_1 \leqslant x \leqslant x_2$ 的一元函数无约束优化问题提供了 fminbnd 函数，函数 fminbnd 的算法基于黄金分割法和二次插值法，它要求目标函数必须是连续函数，并可能只给出局部最优解。fminbnd 的调用格式如下：

（1）x= fminbnd (*fun*,x1,x2)

（2）x= fminbnd (*fun*,x1,x2, *options*)

（3）[x，*fval*]= fminbnd（…）

（4）[x，*fval*，*exitflag*]= fminbnd（…）

（5）[x，*fval*，*exitflag*，*output*]= fminbnd（…）

其中（3）、（4）、（5）的等式右边可选用（1）或（2）的等式右边。

**例 20-1** 求函数 $f=2e^{-2x}\sin(x)$ 在 $0<x<8$ 中的最小值。

**解：**在命令窗口输入：

```
>> f = @(x)2*exp(-2*x).*sin(x);
  fplot(f,[0,8]);          %作图语句
[xmax,ymax]=fminbnd(f, 0, 8)
%输出为
xmax =
    3.6052
ymax =
 -6.6080e-004
```

程序运行后输出如图 20-2 所示的函数图形。

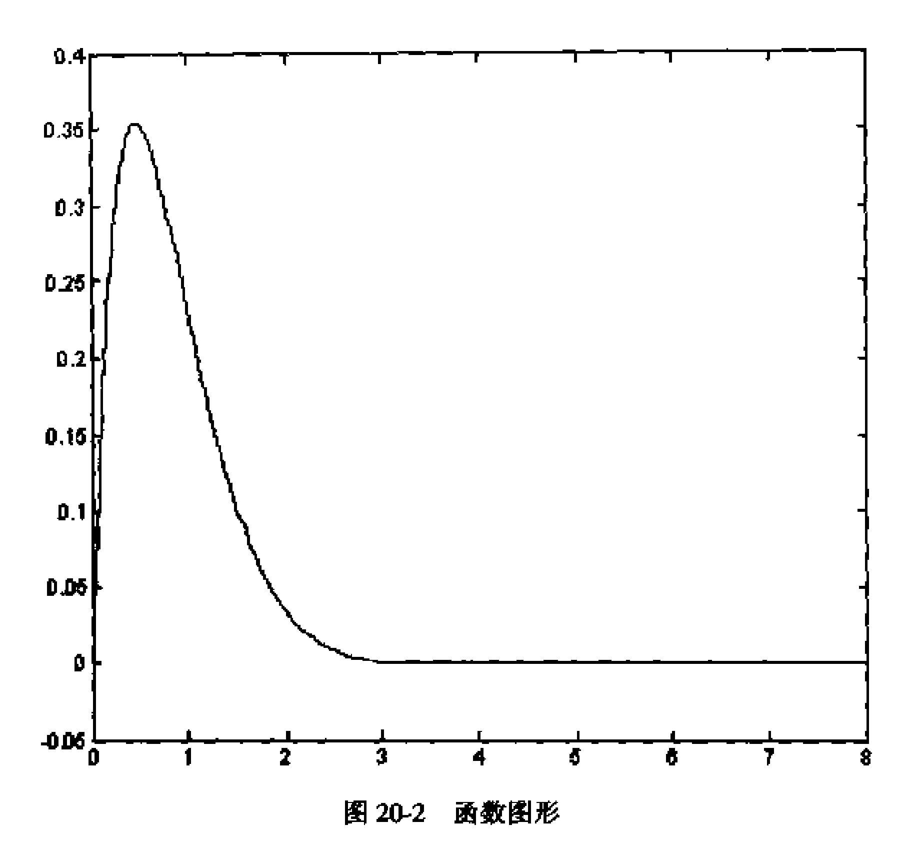



图 20-2 函数图形

由运行结果可知，函数在极点 $x$=3.6052 的极值是 $f$=–6.6080e–004。

### 20.2.2 多元函数无约束优化

在 MATLAB 中，对形如 min $F(X)$ 的多元函数无约束优化问题提供了下列求解函数：

（1）$x$= fminunc（fun, $X$0），或 $x$=fminsearch（fun, $X$0）；

（2）$x$= fminunc（fun, $X$0 , options），或 $x$=fminsearch（fun,$X$0, options）；

（3）[$x$，fval]= fminunc（…），
或[$x$，fval]= fminsearch（…）；

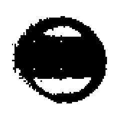
（4）[$x$，fval，exitflag]= fminunc（…），
或[$x$，fval，exitflag]= fminsearch；

（5）[$x$，fval，exitflag，output]= fminunc（…），
或[$x$，fval，exitflag，output]= fminsearch（…）。

其中：fminsearch 是用单纯型法寻优，fminunc 的算法有如下几点说明。

（1）fminunc 为无约束优化提供了大型优化和中型优化算法，由 options 中的参数 LargeScale 控制：LargeScale='on'(默认值)，使用大型算法；LargeScale='off'(默认值)，使用中型算法。

（2）fminunc 为中型优化算法的搜索方向提供了 4 种算法，由 options 中的参数 HessUpdate 控制：HessUpdate='bfgs'（默认值），拟牛顿法的 BFGS 公式；HessUpdate='dfp',



拟牛顿法的 DFP 公式；HessUpdate='steepdesc'，最速下降法。

（3）fminunc 为中型优化算法的步长一维搜索提供了两种算法，由 options 中参数 LineSearchType 控制：LineSearchType='quadcubic'(默认值)，混合的二次和三次多项式插值；LineSearchType='cubicpoly'，三次多项式插值使用 fminunc 和 fminsearch 可能会得到局部最优解。

**例 20-2** 求函数 $f(x)=e^{x_2}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1)$ 在点[-1,1]附近的局部最小点。

**解**：在命令窗口输入：

```
>> f = @(x)exp(x(2))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
x0 = [-1, 1];
x=fminunc(f,x0);
y=f(x)
%输出为
y =
  1.4859e-013
>> x
%输出为
x =
    0.5000   -1.0000
```

可知函数在点[-1,1]附近的点[0.5,-1]处取得局部最小点，极小值为 1.4859e-013。

## 20.3 约束优化问题

约束最优化方法可分为间接法和直接法两大类：

（1）间接法是先将约束优化设计问题转化为一系列的无约束优化设计问题，再调用无约束优化方法来求解。常用的方法有：罚函数法和乘子法等。

（2）直接法是在选取下降方向和下降点时直接判断是否在可行区域内，常用的方法有：约束随机方向法、复合型法等。

上述各种方法都是针对单一的目标函数而设计的，但工程优化设计问题往往是一个多目标优化设计问题。

常见的多目标最优化方法的基本思想是将多目标问题转化为一个或一系列的单目标优化问题，通过求解一个或一系列单目标优化问题来完成多目标优化问题的求解。不同的多目标优化方法有各自不同的转化策略。常用的多目标最优化方法有目标规划法、乘除法、线性加权组合法和功效系数法等。

**例 20-3** 求函数 $f(x)=2x_1^2+4x_2^2-4x_1x_2-6x_1-3x_2$ 在点[1,1]附近，在约束条件下 $\begin{cases}x_1+x_2\leqslant 3\\4x_1+x_2\leqslant 9\\x_1,x_2\geqslant 0\end{cases}$ 的局部最小值。

**解**：约束条件以矩阵的形式表示为：





$$\begin{bmatrix} 1 & 1 \\ 4 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leqslant \begin{bmatrix} 3 \\ 9 \end{bmatrix}$$

在命令窗口输入：

```
>> f = @(x)(2*x(1)^2+4*x(2)^2-4*x(1)*x(2)-6*x(1)-3*x(2));
A=[1,1;4,1];
b=[3;9];
x0=[1;1]
[x,fval]=fmincon(f,x0,A,b)
%输出为
x0 =
     1
     1
x =
   1.9500
   1.0500
fval =
  -11.0250
```

可知函数在点[1,1]附近的点[1.9500,1.0500]处取得局部最小点，极小值为-11.0250。

## 20.4　小结

本章对 MATLAB 的优化工具箱进行了介绍，讲述了常用的无约束优化问题和约束优化问题，为熟练使用优化工具箱进行最优化计算打下基础。

# 第 21 章 微积分

微分运算和积分运算是微积分的基础。微分计算的是一个函数的斜率或梯度，而积分计算的是一个函数所包含的面积。MATLAB 提供了进行微积分计算的函数，用户可以方便地进行微积分运算。

## 21.1 微分

### 21.1.1 符号微分

在 MATLAB 符号函数工具箱中，符号导数由函数 diff 来实现，其调用格式为：

（1）diff(*s*)，没有指定变量和导数阶数，则系统按 findsym 函数指示的默认变量对符号表达式 *s* 求一阶导数；

（2）diff(*s*,'v')，以 *v* 为自变量，对符号表达式 *s* 求一阶导数；

（3）diff(*s*,*n*)，按 findsym 函数指示的默认变量对符号表达式 *s* 求 *n* 阶导数，*n* 为正整数；

（4）diff(*s*,'v',*n*)，以 *v* 为自变量，对符号表达式 *s* 求 *n* 阶导数。

**例 21-1** 计算 $\frac{\mathrm{d}\sin(t^2)}{\mathrm{d}t}$。

**解**：在命令窗口输入：

```
>> syms t
diff(sin(t^2));
>> ans
%输出为
ans =
 2*cos(t^2)*t
```



可知，$\frac{\mathrm{d}\sin(t^2)}{\mathrm{d}t}=2t\cos(t^2)$。

**例 21-2**　计算 $\frac{\mathrm{d}^5t^5}{\mathrm{d}t^5}$。

**解**：在命令窗口输入：

```
>> syms t
diff(t^5,5);
>> ans
%输出为
ans =
120
```

可知，$\frac{\mathrm{d}^5t^5}{\mathrm{d}t^5}=120$。

### 21.1.2　数值微分

在 MATLAB 中，没有直接提供求数值导数的函数，只提供了一个计算其非常粗略的微分的函数。这个函数命名为 diff，它计算数组中元素间的差分，其调用格式为：

（1）***DX***=diff(***X***)，计算向量 ***X*** 的向前差分，$DX(i)=X(i+1)-X(i)$，$i=1,2,\cdots,n-1$；

（2）***DX***=diff(***X***,***n***)，计算 ***X*** 的 $n$ 阶向前差分，例如，diff(***X***,2)=diff(diff(***X***))；

（3）***DX***=diff(***A***,***n***,dim)，计算矩阵 ***A*** 的 $n$ 阶差分，dim=1 时(默认状态)，按列计算差分；dim=2 时，按行计算差分。

**例 21-3**　4 行 4 列的以向量 ***V***=[1,2,3,4]为基的范得蒙矩阵，按列进行差分运算。

**解**：在命令窗口输入：

```
>>V=vander(1:4)
DV=diff(V)        %计算 V 的一阶差分
%输出为
V =
     1     1     1     1
     8     4     2     1
    27     9     3     1
    64    16     4     1
DV =
     7     3     1     0
    19     5     1     0
    37     7     1     0
```

## 21.2　积分

### 21.2.1　符号积分

在 MATLAB 符号函数工具箱中，符号积分由函数 int 来实现，其调用格式为：

（1）int(*s*)，没有指定积分变量和积分阶数时，系统按 findsym 函数指示的默认变量对被积函数或符号表达式 *s* 求不定积分；

（2）int(*s*,*v*)，以 *v* 为自变量，对被积函数或符号表达式 *s* 求不定积分；

（3）int(*s*,*v*,*a*,*b*)，求定积分运算。*a*,*b* 分别表示定积分的下限和上限。该函数求被积函数在区间[*a*,*b*]上的定积分。*a* 和 *b* 可以是两个具体的数，也可以是一个符号表达式，还可以是无穷(inf)。当函数 *f* 关于变量 *x* 在闭区间[*a*,*b*]上可积时，函数返回一个定积分结果。当 *a*,*b* 中有一个是 inf 时，函数返回一个广义积分。当 *a*,*b* 中有一个符号表达式时，函数返回一个符号函数。

**例 21-4** 采用符号积分求 $\int\frac{-2x}{(1+x^2)^2}\mathrm{d}x$。

**解**：在命令窗口输入：

```
>> syms x
int(-2*x/(1+x^2)^2)
 %输出为
ans =
 1/(1+x^2)
```

可知，$\int\frac{-2x}{(1+x^2)^2}\mathrm{d}x=\frac{1}{(1+x^2)}$。

## 21.2.2 数值积分的实现方法

求解定积分的数值方法多种多样，如简单的梯形法、辛普生(Simpson)法、牛顿－柯特斯(Newton-Cotes)法等都是经常采用的方法。

定积分数值方法的基本思想都是将整个积分区间[*a*,*b*]分成 *n* 个子区间[xi,xi+1]，*i*=1,2,…,*n*，其中 x1=*a*，xn+1=*b*。这样求定积分问题就分解为求和问题。

### 1. 梯形法数值积分

MATLAB 给出了采用梯形法求数值积分的 trapz 函数。该函数的调用格式有如下几种：

（1）*T* = trapz(*Y*)

用等距梯形法近似计算 ***Y*** 的积分。若 ***Y*** 是一向量，则 trapz(***Y***)为 ***Y*** 的积分；若 ***Y*** 是一矩阵，则 trapz(***Y***)为 ***Y*** 的每一列的积分；若 ***Y*** 是一多维阵列，则 trapz(***Y***)沿着 ***Y*** 的第一个非单元集的方向进行计算。

（2）*T* = trapz(*X*,*Y*)

用梯形法计算 ***Y*** 在 ***X*** 点上的积分。若 ***X*** 为一列向量，***Y*** 为矩阵，且 size(***Y***,1) = length(***X***)，则 trapz(***X***,***Y***)通过 ***Y*** 的第一个非单元集方向进行计算。

（3）*T* = trapz(*X*,*Y*,dim)或 *T* = trapz(*Y*,dim)

沿着 dim 指定的方向对 ***Y*** 进行积分。若参量中包含 ***X***，则应有 length(***X***)=size(***Y***,dim)。



**例 21-5**　采用梯形法计算定积分 $\int_0^{\pi}\sin(x)\mathrm{d}x$。

**解**：在命令窗口输入：

```
>> X = 0:pi/100:pi;
Y = sin(X);
Z = trapz(X,Y)
%输出为
Z =
    1.9998
```

可知采用梯形法计算 $\int_0^{\pi}\sin(x)\mathrm{d}x = 1.9998$。

**2. 变步长辛普生法数值积分**

MATLAB 给出了采用变步长辛普生法求定积分的 quad 函数。该函数的调用格式为：

[*I*,*n*]=quad('fname',*a*,*b*,tol,trace)

其中 fname 是被积函数名；*a* 和 *b* 分别是定积分的下限和上限；tol 用来控制积分精度，默认时取 tol=0.001；trace 控制是否展现积分过程，若取非 0 则展现积分过程，取 0 则不展现，默认时取 trace=0；返回参数 *I* 即定积分值；*n* 为被积函数的调用次数。

**例 21-6**　采用变步长辛普生法求函数 $f(x)=\mathrm{e}^{-0.4x}\cos(x+\pi/3)$ 在 $[0,3\pi]$ 上的定积分。

**解**：首先建立被积函数文件 func_int.m。

```
function f=func_int(x)
f=exp(-0.4*x).*cos(x+pi/3);
```

然后调用数值积分函数 quad 求定积分，在命令窗口输入：

```
>> [S,n]=quad('func_int',0,3*pi)
%输出为
S =
   -0.5874
n =
    65
```

**3. 牛顿—柯特斯法数值积分**

MATLAB 给出了采用牛顿—柯特斯法求定积分的 quadl 函数。该函数的调用格式为：

*q*=quadl('fname',*a*,*b*)

其中 fname 是被积函数名；*a* 和 *b* 分别是定积分的下限和上限；误差的默认值取 $10^{-6}$。该函数可以更精确地求出定积分的值，且一般情况下函数调用的步数明显小于 quadl 函数，从而保证能以更高的效率求出所需的定积分值。

**例 21-7**　采用牛顿—柯特斯法求函数 $f(x)=\mathrm{e}^{-0.4x}\cos(x+\pi/3)$ 在 $[0,3\pi]$ 上的定积分。

**解**：首先建立被积函数文件 func_int.m。

```
function f=func_int(x)
f=exp(-0.4*x).*cos(x+pi/3);
```

然后调用数值积分函数 quadl 求定积分。

```
>> I=quadl('func_int',0,3*pi)
%输出为
I =
   -0.5874
```

4. 表格形式定义的函数的定积分求法

在 MATLAB 中，对由表格形式定义的函数关系的求定积分问题，采用 trapz(*X*,*Y*)函数。其中向量 *X*,*Y* 定义函数关系 *Y*=*f*(*X*)。

**例 21-8** 采用 trapz 函数计算函数 $y=e^{-x}$ 在区间[1,2.5]之间的定积分，其中函数如图 21-1 所示。

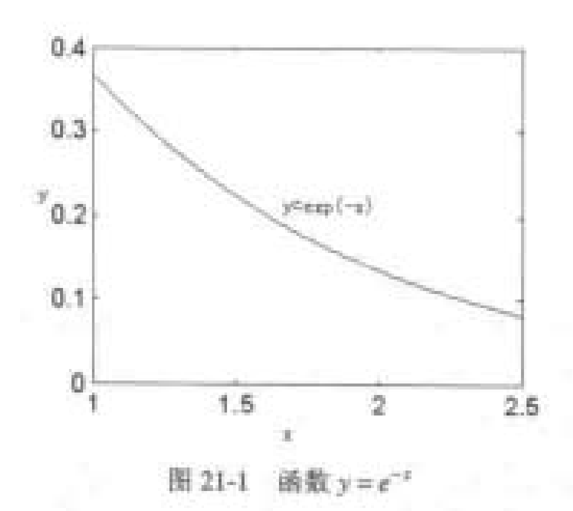



图 21-1 函数 $y=e^{-x}$

**解**：在命令窗口输入：

```
>> x=1:0.01:2.5;
y=exp(-x);        %生成函数关系数据向量
trapz(x,y)
%输出为
ans =
    0.2858
```

### 21.2.3 重积分的实现方法

1. 二重定积分的数值解

MATLAB 提供了直接求出二重定积分的数值解的 dblquad 函数。该函数的调用格式为：

（1）*q*=dblquad(fun,xmin,xmax,ymin,ymax)

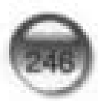
在区域[xmin,xmax, ymin,ymax]上计算二元函数 *z*=*f*(*x*,*y*)的二重积分。输入向量 *x*，标量 *y*，则 *f*(*x*,*y*)必须返回一用于积分的向量。

（2）*q*= dblquad(fun,xmin,xmax,ymin,ymax,tol)

用指定的精度 tol 代替默认精度 $10^{-6}$，再进行计算。

（3）*q*= dblquad(fun,xmin,xmax,ymin,ymax,tol,method)

用指定的算法 method 代替默认算法 quad。method 的取值有@quadl 或用户指定的、与





命令 quad 与 quadl 有相同调用次序的函数句柄。

（4）q=dblquad(fun,xmin,xmax,ymin,ymax,tol,method,p1,p2,⋯)

将可选参数 p1,p2,⋯等传递给函数 fun(x,y,p1,p2,⋯)。若 tol=[]，method=[]，则使用默认精度和算法 quad。

**例 21-9** 计算 $f(x,y)=\mathrm{e}^{-\frac{x^2}{3}}\sin(x^2+2y)$ 在区间$[-1,1]\times[-1,1]$上的二重定积分。

**解：** 首先建立一个函数文件 fxy.m：

```
function f=fxy(x,y)
global ki;
ki=ki+1;                %ki 用于统计被积函数的调用次数
f=exp(-x.^2/3).*sin(x.^2+2*y);
```

然后调用 dblquad 函数求解，在命令窗口输入：

```
>> global ki;ki=0;
I=dblquad('fxy',-1,1,-1,1)
ki
%输出为
I =
    0.4658   %二重定积分值
ki =
   460    %被积函数的调用次数
```

**2．三重定积分的数值解**

MATLAB 提供了直接求出三重定积分的数值解的 triplequad 函数。该函数的调用格式为：

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol)

该函数求 fun(x,y,z)在[xmin,xmax]×[ymin,ymax]×[zmin,zmax]区域上的三重定积分。参数 tol 可以指定，默认值是 1.e–6。

**例 21-10** 计算 $f(x,y,z)=y\sin(x)+z\cos(x)$ 在空间 $[0,\pi]\times[0,1]\times[-1,1]$ 上的三重定积分。

**解：** 在命令窗口输入：

```
>> F = @(x,y,z)y*sin(x)+z*cos(x);
Q = triplequad(F,0,pi,0,1,-1,1);
>> Q
%输出为
Q =
    2.0000
```

可知该三重积分的值为 2。

## 21.3 小结

本章讲述了 MATLAB 中微分运算和积分运算的函数及其使用，对微积分的符号求解和数值求解进行了介绍，并通过简单而具体的实例介绍了 MATLAB 中函数的使用。



# 第 22 章 常微分方程

常微分方程求解是高等数学的基础内容，在实际中有着广泛的应用。MATLAB 提供了求解常微分方程的函数，可以方便地进行常微分方程的求解。

## 22.1 常微分方程符号解

通常，微分方程式描述系统内部变量的变化率如何受系统内部变量和外部激励如输入的影响。当常微分方程式能够解析求解时，可用 MATLAB 的符号工具箱找到精确解。在常微分方程难以获得解析解的情况下，可以方便地在数值上求解。

一阶常微分方程式（first-order ordinary differential equation, ODE）可写为

$$y' = g(x, y)$$

其中 $x$ 为独立变数，而 y 是 $x$ 的函数。它的解是

$$y=f(x,y)$$

该解可以满足 $y'=f'=g(x,y)$，在初始条件 $y(x_0)=y_0$ 下，可以得到惟一解。

MATLAB 常微分方程符号解语法是：

dsolve('equation','condition')

其中 equation 代表常微分方程式即 $y'=g(x,y)$，且须以 Dy 代表一阶微分项 y'，D2y 代表二阶微分项 y''，condition 则为初始条件。

dsolve 的调用格式为：

（1）dsolve('equation')

给出微分方程的解析解，表示为 $t$ 的函数；

（2）dsolve('equation', 'condition')

给出微分方程初值问题的解，表示为 $t$ 的函数；



（3）dsolve('equation', 'v')

给出微分方程的解析解，表示为 $v$ 的函数；

（4）dsolve('equation', 'condition', 'v')

给出微分方程初值问题的解，表示为 $v$ 的函数。

**例 22-1** 计算微分方程 $\frac{dy}{dx}+3xy=xe^{-x^2}$ 的通解。

**解**：在命令窗口输入：

```
>> dsolve('Dy+3*x*y=x*exp(-x^2)')
 %输出为
ans =
 (1/3*exp(-x*(x-3*t))+C1)*exp(-3*x*t)
```

由于系统默认的自变量是 $t$，显然系统把 $x$ 当做常数，把 $y$ 当做 $t$ 的函数求解，输入命令：

```
>> dsolve('Dy+3*x*y=x*exp(-x^2)','x')
  %输出为
ans =
exp(-x^2)+exp(-3/2*x^2)*C1
```

可知，通解为 $y=e^{-x^2}+e^{-\frac{3}{2}x^2}*C_1$，其中 $C_1$ 为常数。

**例 22-2** 计算微分方程 $xy'+2y-e^x=0$ 在初始条件 $y|_{x=1}=2e$ 下的特解。

**解**：在命令窗口输入：

```
>>  dsolve('x*Dy+2*y-exp(x)=0','y(1)=2*exp(1)','x')
   %输出为
ans =
 (exp(x)*x-exp(x)+2*exp(1))/x^2
```

可知特解为 $y=\frac{xe^x-e^x+2e}{x^2}$。

**例 22-3** 求 $y''+2y'+e^x=0$ 的通解。

**解**：在命令窗口输入：

```
>>  dsolve('D2y+2*Dy+exp(x)=0','x')
  %输出为
ans =
 -1/3*exp(x)-1/2*exp(-2*x)*C1+C2
```

可知通解为 $y=-\frac{1}{3}e^x-\frac{1}{2}e^{-2x}*C_1+C_2$，其中 $C_1$，$C_2$ 为常数。

## 22.2 常微分方程数值解

在生产和科研中，所处理的微分方程往往很复杂，且大多得不出一般解。而在实际上对初值问题，一般是要求得到的解在若干个点上满足规定精确度的近似值，或者得到一个



满足精确度要求的便于计算的表达式。

对于常微分方程：$\begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \end{cases}$，其数值解是指由初始点 $x_0$ 开始的若干离散的 $x$ 值处，即对 $x_0 < x_1 < x_2 < \cdots < x_n$，求出准确值 $y(x_1), y(x_2), \cdots, y(x_n)$ 的相应近似值 $y_1, y_2, \cdots, y_n$。因此，研究常微分方程的数值解法是十分必要的。

在求常微分方程数值解方面，MATLAB 具有丰富的函数，将其统称为 solver，其一般格式为：

$$[T,Y]=\text{solver}(\text{odefun},tspan,y0)$$

该函数表示在区间 $tspan=[t_0,t_f]$上，用初始条件 y0 求解显式常微分方程 $y' = f(t,y)$。

odefun 为显式常微分方程 $y' = f(t,y)$ 中的 $f(t,y)$，*tspan* 为求解区间，要获得问题在其他指定点 $t_0,t_1,t_2,\cdots$ 上的解，则令 $tspan=[t_0,t_1,t_2,\cdots,t_f]$（要求 $t_i$ 单调），y0 为初始条件。

solver 为命令 ode45，ode23，ode113，ode15s，ode23s，ode23t，ode23tb 之一，其中 ode45，ode23，ode113 属于非刚性 ODE 类型，这些命令的特点如表 22-1 所示；ode15s，ode23s，ode23t，ode23tb 属于刚性 ODE 类型，这些命令的特点如表 22-2 所示。

**表 22-1　非刚性 ODE 求解命令**

| 求解器 Solver | 特　　点 | 说　　明 |
|---|---|---|
| ode45 | 一步算法；4，5 阶 Runge-Kutta 方程；累计截断误差达$(\triangle x)^3$ | 大部分场合的首选算法 |
| ode23 | 一步算法；2，3 阶 Runge-Kutta 方程；累计截断误差达$(\triangle x)^3$ | 使用于精度较低的情形 |
| ode113 | 多步法；Adams 算法；高低精度均可到 $10^{-3}$~$10^{-6}$ | 计算时间比 ode45 短 |

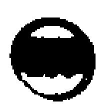
**表 22-2　刚性 ODE 求解命令**

| 求解器 Solver | 特　　点 | 说　　明 |
|---|---|---|
| ode23t | 采用梯形算法 | 适度刚性情形 |
| ode15s | 多步法；Gear's 反向数值微分；精度中等 | 若 ode45 失效时，可尝试使用 |
| ode23s | 一步法；2 阶 Rosebrock 算法；低精度 | 当精度较低时，计算时间比 ode15s 短 |
| ode23tb | 梯形算法；低精度 | 当精度较低时，计算时间比 ode15s 短 |

函数 ode45 与 ode23 是常使用的求解方法，函数 ode45 的使用与 ode23 完全一样。两个函数的差别在于必须与所用的内部算法相关。两个函数都运用了基本的龙格-库塔（Runge-Kutta）数值积分法的变形。

ode23 运用一个组合的 2/3 阶龙格-库塔-芬尔格（Runge-Kutta-Fehlerg）算法，而 ode45 运用组合的 4/5 阶龙格-库塔-芬尔格算法。一般地，ode45 可取较多的时间步，因此，要保持与 ode23 相同误差时，在 $t_0$ 和 $t_f$之间可取较少的时间步。然而，在同一时间，ode23 每时间步至少调用 3 次，而 ode45 每时间步至少调用 6 次。

正如使用高阶多项式内插常常得不到最好的结果一样，ode45 也不总是比 ode23 好。如果 ode45 产生的结果，对作图间隔太大，则必须在更细的时间区间，对数据进行内插，





比如用函数 interp1。这个附加时间点会使 ode23 更有效。作为一条普遍规则，在所计算的导数中，如有重复的不连续点，为保持精度致使高阶算法减少时间步长，这时低阶算法更有效。

求解具体 ODE 的基本过程如下：

（1）根据问题所属学科中的规律、定律、公式，用微分方程与初始条件进行描述。

$$F(y, y', \cdots, y^{(n-1)}, t) = 0$$

$$y(0) = y_0, y'(0) = y_1, \cdots, y^{(n-1)}(0) = y_{n-1}$$

写为向量的形式为 $y = [y; y(1); y(2); \cdots; y(m-1)]$，$n$ 与 $m$ 可以不等。

（2）运用数学中的变量替换：$y_n = y(n-1), y_{n-1} = y(n-2), \cdots, y_2 = y_1 = y$，把高阶（大于 2 阶）的方程（组）写成一阶微分方程组：

$$\boldsymbol{y}' = \begin{bmatrix} y_1' \\ y_2' \\ \cdots \\ y_n' \end{bmatrix} = \begin{bmatrix} f_1(t, y) \\ f_2(t, y) \\ \cdots \\ f_n(t, y) \end{bmatrix}$$

相应的初始条件为：

$$\boldsymbol{y}_0 = \begin{bmatrix} y_1(0) \\ y_2(0) \\ \cdots \\ y_n(0) \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_n \end{bmatrix}$$

（3）根据（1）与（2）的结果，编写能计算导数的 M-函数文件 odefile。

（4）将文件 odefile 与初始条件传递给求解器 Solver 中的一个，运行后就可得到 ODE 的、在指定时间区间上的解列向量 $y$（其中包含 $y$ 及不同阶的导数）。

**例 22-4** 求描述某非刚性体的运动方程的微分方程 $\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = 0.51 y_1 y_2 \end{cases}$，其初始条件为 $\begin{cases} y_1(0) = 0 \\ y_2(0) = 1 \\ y_3(0) = 1 \end{cases}$。

**解**：首先编写函数文件 rigid.m：

```
function dy = rigid(t,y)
dy = zeros(3,1);    % 行向量
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
```

再在命令窗口中执行：

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
[T,Y] = ode45(@rigid,[0 12],[0 1 1],options);
```

```
plot(T,Y(:,1),'-',T,Y(:,2),'-.',T,Y(:,3),'.')
```

输出的图形结果为图 22-1 所示。

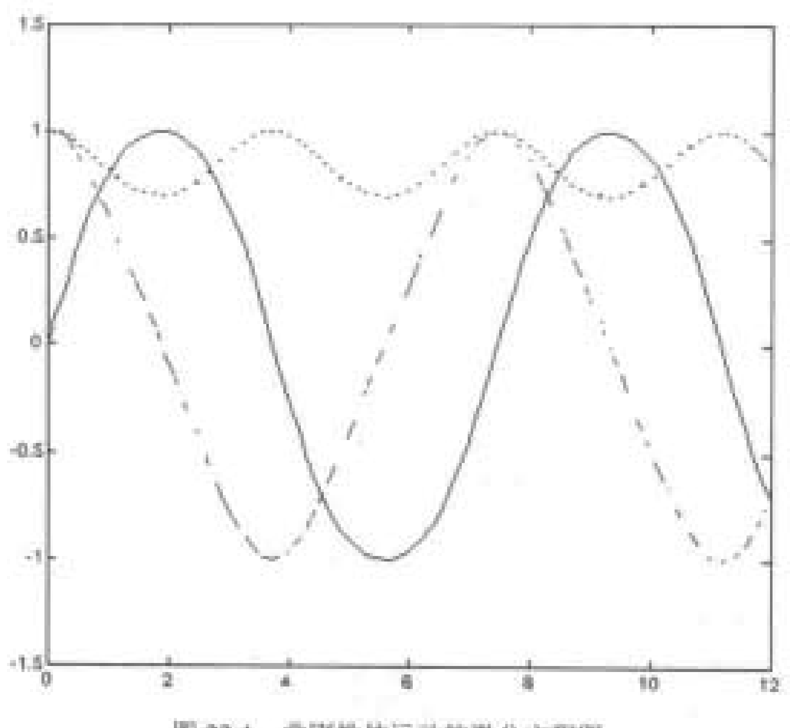



图 22-1　非刚性体运动的微分方程图

## 22.3　小结

本章讲述了常微分方程的符号解和数值解，介绍了 MATLAB 中相应的求解函数，并通过实例讲述了常微分方程的求解。

# 第 23 章 二维图形

数据可视化是 MATLAB 的一项重要功能。通过数据可视化的方法，工程科研人员可以对自己的样本数据的分布、趋势特性有一个直观的了解。MATLAB 中实现数据可视化的方法包括二维绘图和三维绘图。

本章主要讲解绘制二维平面图形实现数据可视化的方法，从 MATLAB 的图形窗口的界面入手，首先概述图形窗口界面提供的基本功能，使读者熟悉 MATLAB 图形显示和处理环境，然后深入讲解 MATLAB 中的基本绘图函数、图形标注函数和特殊绘图函数，最后介绍图形窗口一些高级功能的应用。

## 23.1 MATLAB 图形窗口概述

MATLAB 中提供了丰富的绘图函数和绘图工具，这些函数或者工具的输出都显示在 MATLAB 命令窗口外的一个图形窗口中，图 23-1 就是一个典型 MATLAB 图形窗口。

和很多 Windows 标准应用程序窗口类似，MATLAB 图形窗口由标题栏、菜单栏、工具条和图形区组成。

标题栏左侧显示该图形的文件名，右侧是图形最大、最小化以及关闭按钮。

菜单栏包括文件（File）、编辑（Edit）、视图（View）、插入（Insert）、工具（Tools）、桌面（Desktop）、窗口（Window）和帮助（Help）菜单。菜单栏右边的箭头可以把图形窗口显示在 MATLAB 桌面中。

默认视图下的图形工具条只包括以下功能的工具按钮：新建文件、打开文件、保存文件、打印文件；图形编辑模式开关；放大、缩小、平移、旋转；数据点标记；颜色条、图例；隐藏绘图工具、显示绘图工具。用户也可以通过单击视图菜单（View）下的子菜单打

开照相机工具条（Camera Toolbar）和图形编辑工具条（Plot Edit Toolbar）。

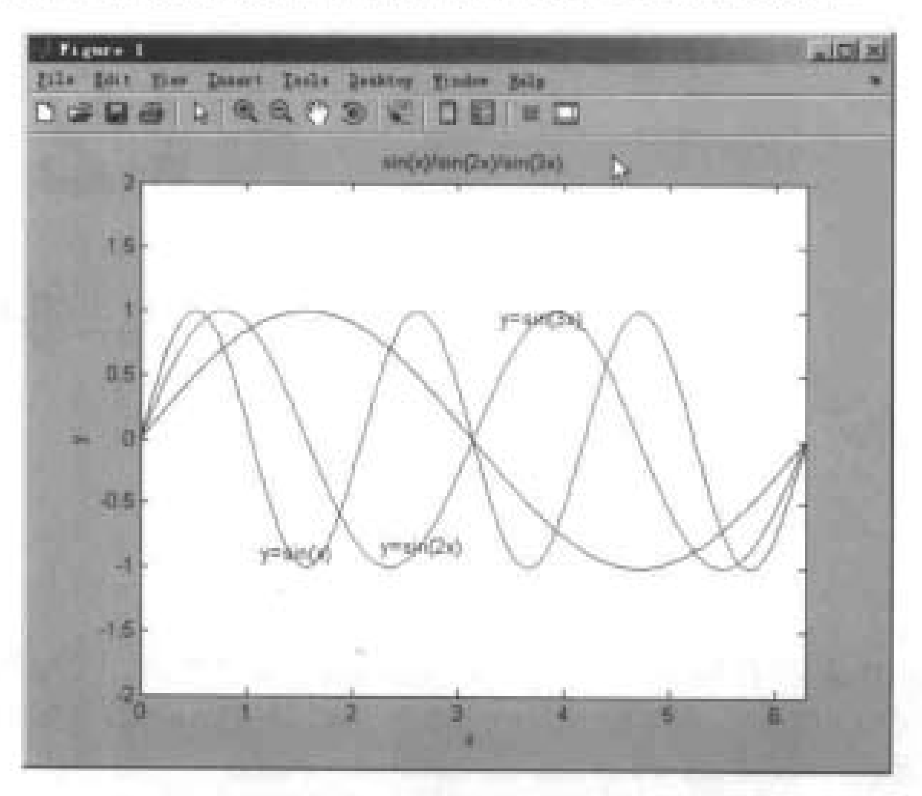



图 23-1 MATLAB 图形窗口

图形区用于显示通过绘图函数或工具绘制的目标图形。一幅典型的二维图形包括标题（如图 23-1 中的 sin(x)/sin(2x)/sin(3x)）、坐标轴、函数图形（如图 23-1 中三条颜色不同的曲线）、标注（如图 23-1 中 y=sin(x)、y=sin(2x)、y=sin(3x)这些文字标注）。从图 23-1 中容易看到，MATLAB 通过线型和颜色来区分多条曲线。

大多数情况下，用户需要自己编写 MATLAB 代码来实现个人数据的可视化，这需要用户熟悉掌握 MATLAB 中种类繁多的绘图函数，本章接下来的几节重点讲述这些绘图函数的用法。另一种更简单易用的方法是使用 MATLAB 提供的图形绘制工具，通过单击视图菜单（View）下的图形面板子菜单（Figure Palette）就可以打开图形面板，利用各种图形面板下的各种绘图工具对工作区变量进行绘图是很方便的，而且图形面板中还提供了丰富的图形标注工具。通过工具绘图的方法，在本章的最后部分会专门讲述。

通过绘图工具绘制的图形也可以转换为 MATLAB 函数文件，只需要单击文件菜单（File）中的子菜单项产生 M-文件（Generate M-File…），就可以把绘图结果保存为 MATLAB 函数代码，这对于学习 MATLAB 中的绘图函数、标注方法以及绘图方法的重复利用、复杂标注的简易实现等都是十分实用的，该菜单位置如图 23-2 所示。

例 23-1 就是通过图 23-1 产生的绘图函数代码，其中有许多本章后续内容将要讲解的函数，读者可以先试着阅读一下，想想各条指令的含义。



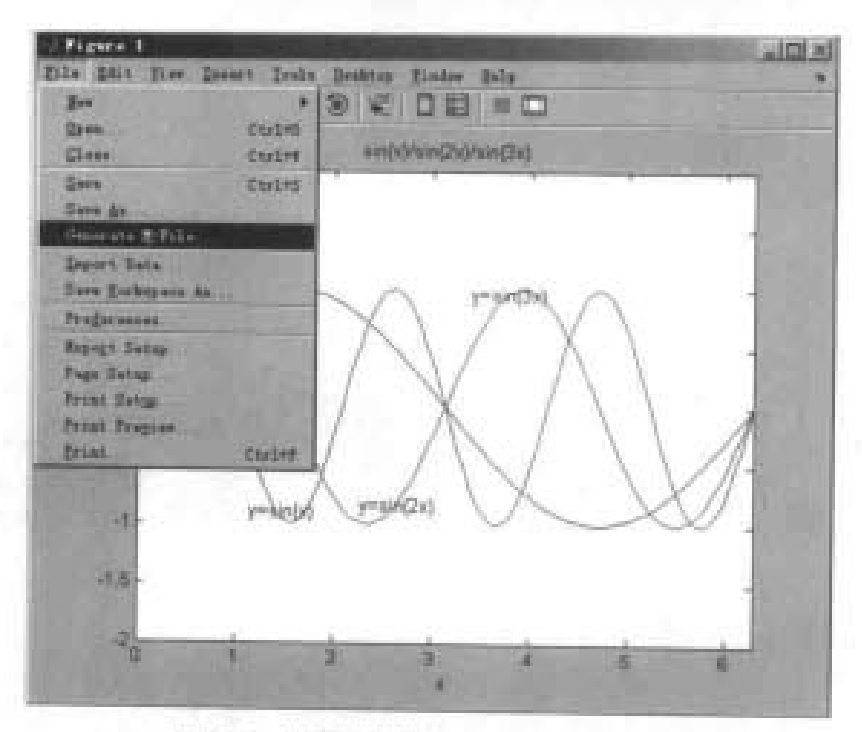



图 23-2　将图形转换为 MATLAB 函数文件

**例 23-1**　MATLAB 绘图函数实例。

**解：**在命令窗口输入：

```
function createfigure(x1, y1)
%CREATEFIGURE(X1,Y1)
%  X1:  vector of x data
%  Y1:  matrix of y data

%  Auto-generated by MATLAB on 16-Feb-2006 14:38:25

%% Create figure
figure1 = figure('PaperPosition',[0.6345 6.345 20.3 15.23], 'PaperSize',
[20.98 29.68]);

%% Create axes
axes1 = axes('Parent',figure1);
axis(axes1,[0 6.283 -2 2]);
title(axes1,'sin(x)/sin(2x)/sin(3x)');
xlabel(axes1,'x');
ylabel(axes1,'y');
box(axes1,'on');
hold(axes1,'all');

%% Create multiple lines using matrix input to plot
plot1 = plot(x1,y1);

%% Create text
text1 = text(...
  'Position',[1.136 -0.9532 0],...
  'String','y=sin(x)',...
```

```
  'VerticalAlignment','baseline', ...
  'Parent',axes1);

%% Create text
text2 = text(...
  'Position',[2.266 -0.8947 0], ...
  'String','y=sin(2x)', ...
  'VerticalAlignment','baseline', ...
  'Parent',axes1);

%% Create text
text3 = text(...
  'Position',[3.395 0.8947 0], ...
  'String','y=sin(3x)', ...
  'VerticalAlignment','baseline', ...
  'Parent',axes1);
```

## 23.2 基本绘图指令

本节介绍MATLAB中命令窗口下的基本绘图流程和各绘图步骤中用到的MATLAB函数。这些命令行下的绘图指令是MATLAB绘图的基础和精髓，读者一定要仔细体会和掌握本节中的每一个函数和实例。

### 23.2.1 基本绘图流程

MATLAB中绘制一个典型的图形文件，需要经过以下7个步骤：

（1）数据准备；

（2）设置当前绘图区；

（3）绘图；

（4）设置图形中曲线和标记点格式；

（5）设置坐标轴和网格线属性；

（6）标注图形；

（7）保存和导出图形。

表23-1以例23-1的MATLAB函数文件为基础，说明这基本的7步对应的具体代码。

表23-1 基本绘图流程

| 绘图流程 | M-代码举例（以例23-1为基础） |
|---|---|
| 1．数据准备 | 准备好绘图需要的 $x$ 变量和 $y$ 变量数据<br>（注：图23-1的数据准备代码为：<br>x1=0:0.02*pi:2*pi;<br>y1=[sin(x1);sin(2*x1);sin(3*x1)];） |
| 2．设置当前绘图区 | figure1 = figure('PaperPosition',[0.6345 6.345 20.3 15.23],'PaperSize',[20.98 29.68]);<br>在指定的位置创建新的绘图窗口，并自动以此窗口的绘图区为当前绘图区 |



续表

| 绘图流程 | M-代码举例（以例 23-1 为基础） |
|---|---|
| | （注：在子图绘制时需要指定当前绘图区在整个绘图区中的具体位置，如 figure(1);subplot(2,2,1) 则是创建新图形窗口，并绘制 2 行 2 列个子图，当前绘图位置为第 1 行第 1 列。这在本节最后部分将会讲述。） |
| 3．绘图 | axes1 = axes('Parent',figure1);<br>hold(axes1,'all');<br>plot1 = plot(x1,y1);<br>创建坐标轴，指定叠加绘图模式，绘制函数曲线 |
| 4．设置图形中曲线和标记点格式 | 用 set 函数设置图形中的线宽、线型、颜色和标记点的形状、大小、颜色等，例 23-1 代码中采用了 MATLAB 默认设置<br>（注：这部分内容可以参考本节的 23.2.3。） |
| 5．设置坐标轴和网格线属性 | axis(axes1,[0 6.283 -2 2]);<br>将坐标轴的范围设置在指定曲线<br>（注：经常还需要指定坐标轴标度点、图形网格线等。） |
| 6．标注图形 | title(axes1,'sin(x)/sin(2x)/sin(3x)');<br>xlabel(axes1,'x');<br>ylabel(axes1,'y');<br>box(axes1,'on');<br>text1 = text(…<br>'Position',[1.136 -0.9532 0], …<br>'String','y=sin(x)', … |
| | 'VerticalAlignment','baseline', …<br>'Parent',axes1);<br>text2 =…;<br>text3=…;<br>在图形中添加标题、坐标轴标注、文字标注等 |
| 7．保存和导出图形 | 按指定文件格式、属性保存或导出图形，<br>如：print -depsc -tiff -r200 myplot |

对照表 23-1 和例 23-1，可见所谓的 7 个步骤的顺序也不是完全固定，尤其是其中对图形进行修饰标注的 4，5，6 步骤，完全可以改变顺序。

另外，MATLAB 中对于图形中的曲线和标记点格式有默认的设置，这在一般情况下是可以满足使用者需要的，因此对于只是想大概查看一下数据分布的用户，只需要进行第 1，3 步工作就可以了。

## 23.2.2 基本绘图函数

MATLAB 中最基本的二维绘图函数是：直角坐标系下的简单画线函数 line、核心绘图函数 plot 和极坐标系下的绘图函数 polar。

line 函数的常用语法格式为：



line(*X*,*Y*)

其中 *X*，*Y* 都是一维数组，line(*X*,*Y*)能够把(*X*(*i*),*Y*(*i*))代表的各点用线段依次连接起来，从而绘制出一条折线。

**例 23-2** 简单画线函数 line。

**解**：在命令窗口输入：

```
>> x=0:0.4*pi:2*pi;
>> y=sin(x);
>> line(x,y)     %如图 23-3 所示
```

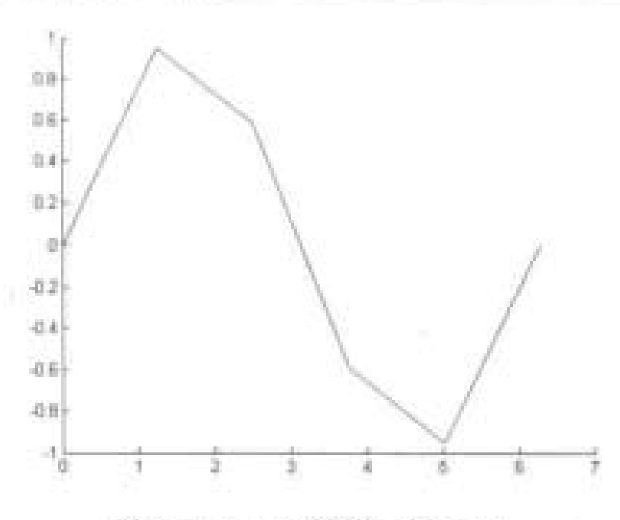


图 23-3 line 函数画线（例 23-2）

plot 函数是 MATLAB 中最核心的二维绘图函数，它有多种语法格式，可以实现多种功能。

plot(*Y*)是 plot 最简单的用法。当 *Y* 是一维数组时，plot(*Y*)是把(*i*,*Y*(*i*))各点依次连接起来，其中 *i* 的取值范围从 1 到 length(*Y*)。当 *Y* 是普通的二维数组时，相当于对 *Y* 的每一列进行 plot(*Y*(:,*i*))画线，并把所有的折线累叠绘制在当前坐标轴下。

plot 最常用的语法格式是接受两个参数的 plot(*X*,*Y*)。

当 *X* 和 *Y* 都是一维数组时，功能和 line(*X*,*Y*)类似；但 plot 函数中的 *X* 和 *Y* 也可以是一般的二维数组，这时候就是对 *X* 和 *Y* 的对应列画线。

特别的，当 *X* 是一个向量，*Y* 是一个在某一方向和 *X* 具有相同长度的二维数组时，plot(*X*,*Y*)则是对 *X* 和 *Y* 的每一行（或列）画线。

plot 函数的这种用法还可以拓展为 plot(*X*1,*Y*1,*X*2,*Y*2,…,*X*n,*Y*n)，这样就可以对多组变量同时进行绘图了，对于每一组变量，其意义同前所述。

**例 23-3** plot 函数应用。

**解**：在命令窗口输入：

```
>> x=0:0.4*pi:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> y3=sin(x-0.1*pi);
```

```
>> y4=cos(x+0.1*pi);
>> plot(y1) %如图 23-4 所示（注意与图 23-3 的横轴的差别）
>> plot(x,y1)   %同图 23-3
>> plot(x,[y1;y2;y3;y4])    %如图 23-5 所示
>> plot(x,y1,x,y2,x,y3,x,y4)    %同图 23-5
```

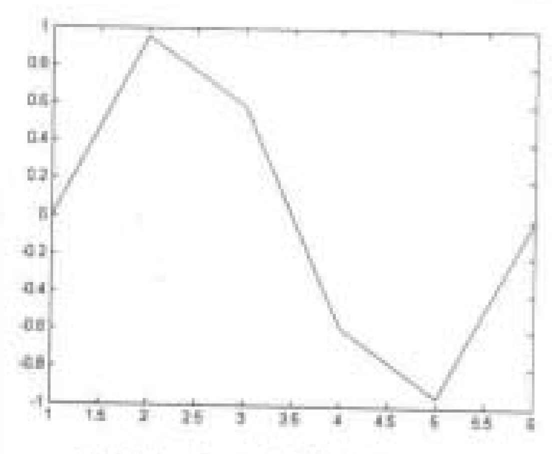


图 23-4 plot(y1)画线结果（例 23-3）

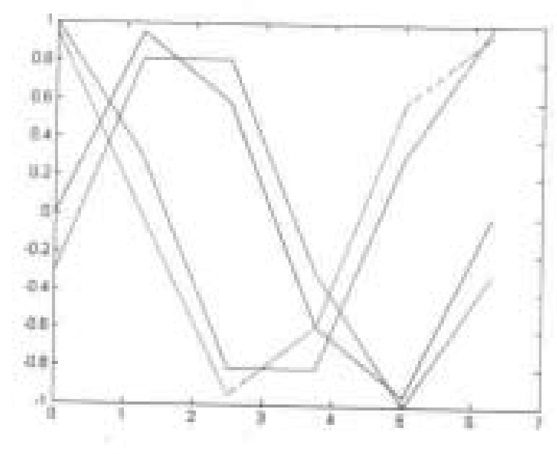


图 23-5 多组数据的 plot 结果（例 23-3）

从图 23-5 可以看到，多组数据绘图时，MATLAB 默认通过不同的颜色区分各条曲线。实际上，plot 函数绘图指令中也可以设置各条曲线的颜色、线型等属性，这时候 plot 函数对应的语法格式为：

plot(*X*1,*Y*1,LineSpec,…)

其中 LineSpec 就是一个指定曲线颜色、线型等特征的字符串。这在下一小节中将专门讲解。

对应于直角坐标系下的绘图函数 line 和 plot，在极坐标系下，MATLAB 也提供了基本的绘图函数 polar。

polar 函数常用的格式有两种：

（1）polar(theta,rho)

（2）polar(theta,rho,LineSpec)

其功能类似于 plot 函数，需要注意的是 theta 和 rho 也可以是普通的二维数组，但 polar 不能接受多对参数输入。

下面对第一种语法格式以实例说明。第二种语法格式中的曲线属性设置可以参考下一小节的例子。

**例 23-4** 极坐标绘图函数 polar。

**解**：在命令窗口输入：

```
>> theta=0:0.05*pi:2*pi;
>> r1=sin(theta);
>> r2=cos(theta);
>> polar(theta,r1,theta,r2)
??? Error using ==> polar
```

```
Too many input arguments.

>> polar(theta,[r1;r2])
??? Error using ==> polar
THETA and RHO must be the same size.

>> polar([theta' theta'],[r1' r2'])
```

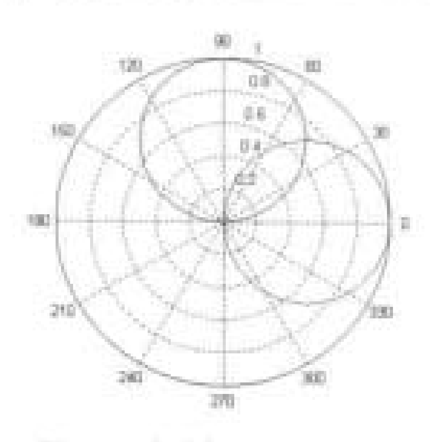



图 23-6 极坐标绘图（例 23-4）

## 23.2.3 设置函数曲线格式和标记点格式

如前所述，plot 函数采用 plot(X1,Y1,LineSpec,…)的语法格式时，可以通过字符串 LineSpec 指定曲线的线型、颜色以及数据点的标记类型。这在突出显示原始数据点和个性化区分多组数据的时候是十分有用的。

例如'-.or'就表示采用点划线，数据点用圆圈标记，颜色都设为红色。需要注意的是，当指定了数据点标记类型，但不指定线型时，则表示只标记数据点，而不进行连线绘图。

MATLAB 默认是用颜色区分多组曲线，但在只能黑白打印或显示的情况下，个性化的设置曲线线型就成了惟一的区分方法。

表 23-2 列出了 MATLAB 中可供选择的曲线线型、颜色和标记点类型。

表 23-2 LineSpec 可选字符串列表

| 线型 | | 颜色 | | 数据点标记类型 | |
|---|---|---|---|---|---|
| 标识符 | 意义 | 标识符 | 意义 | 标识符 | 意义 |
| - | 实线 | r | 红色 | + | 加号 |
| -. | 点划线 | g | 绿色 | o | 圆圈 |
| -- | 虚线 | b | 蓝色 | * | 星号 |
| : | 点线 | c | 蓝绿色 | . | 点 |
| | | m | 洋红色 | x | 交叉符号 |
| | | y | 黄色 | square（或 s） | 方格 |
| | | k | 黑色 | diamond（或 d） | 菱形 |

续表

| 线　型 | 颜　色 | | 数据点标记类型 | |
|---|---|---|---|---|
| | w | 白色 | ^ | 向上的三角形 |
| | | | v | 向下的三角形 |
| | | | > | 向左的三角形 |
| | | | < | 向右的三角形 |
| | | | pentagram（或 p） | 五边形 |
| | | | hexagram（或 h） | 六边形 |

**例 23-5**　曲线格式和标记点类型设置。

**解**：在命令窗口输入：

```
>> x=0:0.1*pi:2*pi;
>> y3=sin(x).*cos(3*x);
>> y1=sin(x);
>> y2=cos(3*x);
>> plot(x,y1,'ob',x,y2,'--dc',x,y3,':vr')  %第一组数据只标记数据点
```

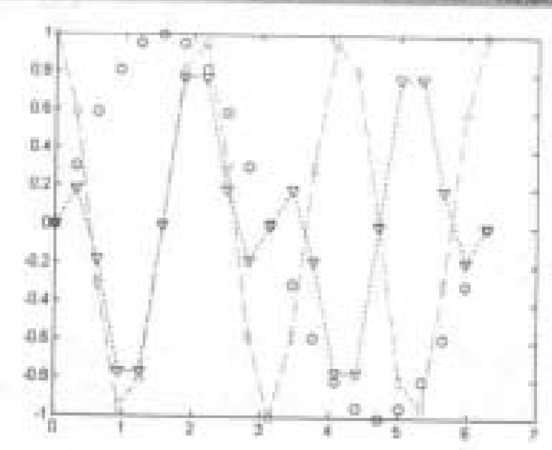



图 23-7　曲线格式和标记点类型设置（例 23-5）

除了上述属性设置外，还可以在 plot 绘图的同时设置曲线线宽、标记点大小，标记点边框颜色和标记点填充颜色等。这些需要通过 plot(…,'PropertyName',PropertyValue,…)这样的语法格式来实现。

其中可供选择的 PropertyName 如表 23-3 所示。

表 23-3　绘图命令中可选的 PropertyName

| PropertyName | 意　义 | 选　项 |
|---|---|---|
| LineWidth | 线宽 | 数值，如 0.5，1，2.5 等，单位为 points |
| MarkerEdgeColor | 标记点边框线条颜色 | 颜色字符，如'g'，'b'，'k'等 |
| MarkerFaceColor | 标记点内部区域填充颜色 | 颜色字符 |
| MarkerSize | 标记点大小 | 数值，单位为 points |

**例 23-6**　线宽和标记点格式设置。

**解**：在命令窗口输入：

```
>> x=-5:0.5:5;
>> y=5.*exp(-abs(x)).*sin(x);
>> plot(x,y,'--hr','LineWidth',1.5, …
'MarkerEdgeColor','b','MarkerFaceColor','m','MarkerSize',10)
```

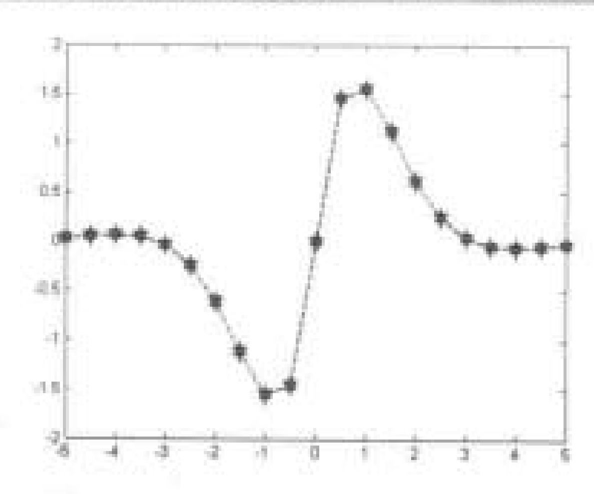


图 23-8　线宽和标记点格式设置（例 23-6）

### 23.2.4　子图绘制

有的时候，为了便于对比或者节省绘图空间，用户需要在同一个绘图窗口下建立多个子图，即建立多个坐标系并在各坐标系中分别绘图，这时候就需要用到 subplot 函数。

subplot 最常用的语法格式为：

subplot(*m*,*n*,*i*)

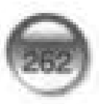
这表示在当前绘图区中建立 *m* 行 *n* 列个绘图子区，并在编号为 *i* 的位置上建立坐标系，并设置该位置为当前绘图区。绘图子区的编号优先从顶行开始，然后是第二行，第三行……

例如，subplot(3,5,9)表示在当前绘图区中建立 3 行 5 列的绘图子区，并在第 2 行，第 4 列的位置建立坐标系准备绘图。

**例 23-7**　子图绘制。

**解**：在命令窗口输入：

```
>> x=-3:0.1:3;
>> y1=x;
>> y2=x.^2;
>> y3=x.^3;
>> y4=x.^4;
>> subplot(2,2,1)
>> plot(x,y1)
>> title('y1=x')        %给当前图形添加标题，下同
>> subplot(2,2,2)
>> plot(x,y2)
>> title('y2=x^2')
>> subplot(2,2,3)
>> plot(x,y3)
>> title('y3=x^3')
```





```
>> subplot(2,2,4)
>> plot(x,y4)
>> title('y4=x^4')
```

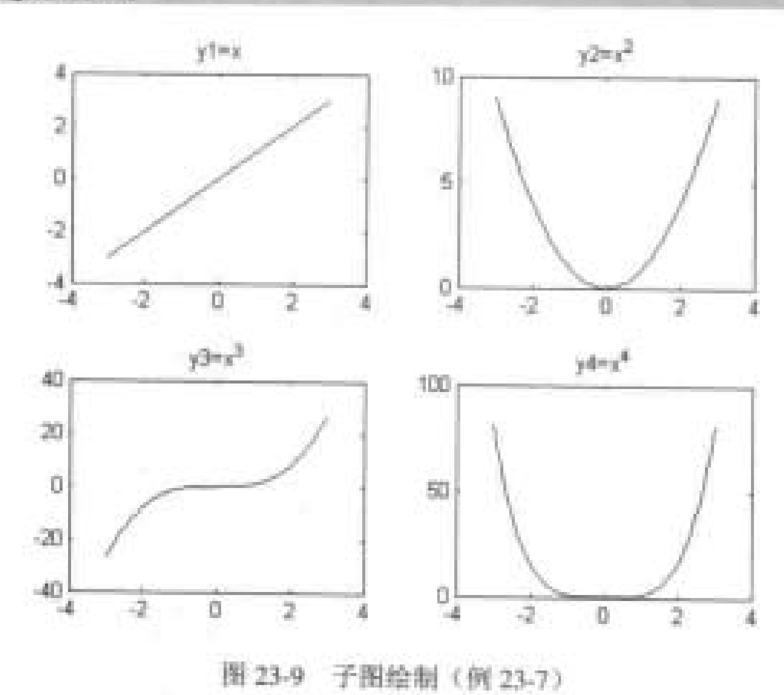



图 23-9　子图绘制（例 23-7）

### 23.2.5　叠加绘图模式

有些情况下，用户需要在已经绘制好的图形上叠加绘制新的图形。MATLAB 提供了开关指令 hold，可以开启或者关闭图形窗口的叠加绘图模式。

单独使用 hold 可以切换当前的绘图叠加模式，将当前绘图窗口的叠加模式从 on 改变为 off，或者从 off 改变到 on。

hold on 或 hold off 则是明确指定当前绘图窗口叠加绘图模式的开关状态。

hold all 不但实现 hold on 的功能，使得当前绘图窗口的叠加绘图模式打开，而且使新的绘图指令依然循环初始设置的颜色循环序和线型循环序。

当 MATLAB 执行到某一条绘图指令时，如果没有图形窗口存在，则 MATLAB 会新建一个图形窗口，并以新建的图形窗口为当前图形窗口绘图；如果有图形窗口已经存在，则该绘图指令会以最后被激活（最后新建、最后被鼠标点击等）的图形窗口为当前图形窗口进行绘图。

如果当前图形窗口的叠加绘图模式关闭时，则新执行的绘图指令会覆盖当前图形窗口中已有的图形，只显示最后的绘图指令的执行结果；而如果当前图形窗口的叠加绘图模式开启时，则新执行的绘图指令绘制的函数曲线或数据点会叠加在原来已经有的图形上。

例 23-8　叠加绘图模式。

**解**：在命令窗口输入：

```
>> x=-5:5;
>> y1=randn(size(x));
```

```
>> y2=normpdf(x);
>> subplot(2,1,1)
>> hold
Current plot held
>> hold %切换子图 1 的叠加绘图模式到关闭状态
Current plot released
>> plot(x,y1,'b')
>> plot(x,y2,'r')   %新的绘图指令冲掉了原来的绘图结果
>> title('hold off mode')
>> subplot(2,1,2)
>> hold on  %打开子图 2 的叠加绘图模式
>> plot(x,y1,'b')
>> plot(x,y2,'r')   %新的绘图结果叠加在原来的图形中
>> title('hold on mode')
```

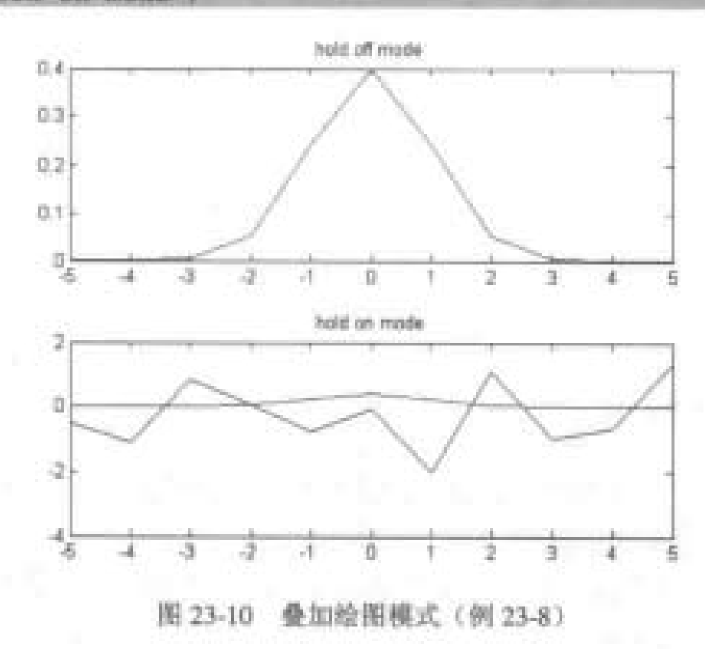



图 23-10　叠加绘图模式（例 23-8）

### 23.2.6　设置坐标轴和网格线

通过 subplot，plot，hold 等函数画出基本的图形之后，就要对图形进行必要的修饰了，这包括坐标轴设置和各种图形标注。本小节介绍坐标轴设置，图形标注的内容见本章 23.3 节。

MATLAB 中对坐标轴的设置包括设置坐标轴范围、标度和纵横比。

坐标轴范围有四种设置模式：

（1）axis([xmin xmax ymin ymax])可以设置坐标轴范围在指定的区间；

（2）axis auto 将当前绘图区的坐标轴范围设置为 MATLAB 自动调整的区间；

（3）axis manual 冻结当前坐标轴范围，以后叠加绘图都在当前坐标范围内显示；

（4）axis tight 采用紧密模式设置当前坐标轴范围，即以用户数据范围为坐标轴范围。

坐标轴比例有三种模式：

（1）axis equal 设置当前坐标轴的横纵轴具有相同的单位长度，即等比例坐标轴；

（2）axis square 以当前坐标轴范围为基础，将坐标轴区域调整为方格形；





（3）axis normal 自动调整纵横轴比例，使当前坐标轴范围内的图形显示达到最佳效果。

特别要提出的是，设置坐标轴范围的选项和设置坐标轴比例的选项可以在 axis 函数中联合使用。MATLAB 绘图是默认的坐标轴设置为 axis auto normal。

**例 23-9**　坐标轴范围和比例设置（M-file）。

**解**：在命令窗口输入：

```
%Ex23-09 axis example
clear
clc
t=0:0.01*pi:2*pi;
x=sin(t);
y=cos(t);
for i=1:9
   subplot(3,3,i)
   plot(x,y)
end
subplot(3,3,1)
axis auto normal
title('axis auto normal')
subplot(3,3,2)
axis([-2 2 -1.5 1.5])
axis normal
title('axis [] normal')
subplot(3,3,3)
axis tight normal
title('axis tight normal')
subplot(3,3,4)
axis auto square
title('axis auto square')
subplot(3,3,5)
axis([-2 2 -1.5 1.5])
axis square
title('axis [] square')
subplot(3,3,6)
axis tight square
title('axis tight square')
subplot(3,3,7)
axis auto equal
title('axis auto equal')
subplot(3,3,8)
axis([-2 2 -1.5 1.5])
axis equal
title('axis [] equal')
subplot(3,3,9)
axis tight equal
title('axis tight equal')
```

另外，还可以通过 MATLAB 命令设置坐标轴的显示刻度。这主要是通过 set 命令来进行。下面通过实际例子说明设置坐标轴的显示刻度的方法。

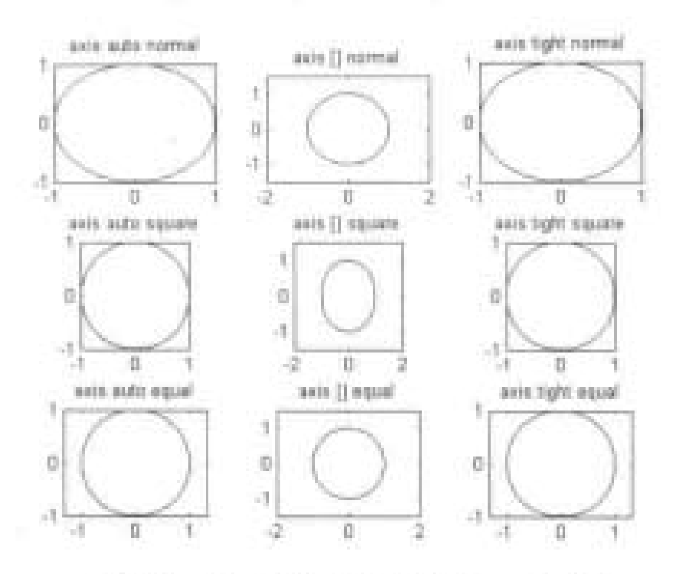



图 23-11　坐标轴范围和比例设置（例 23-9）

**例 23-10**　设置坐标轴显示刻度。

**解**：在命令窗口输入：

```
>> x=0:0.1:6;
>> y=sin(x).*cos(3*x);
>> plot(x,y)
>> set(gca,'XTick',[0 0.5*pi pi 1.5*pi 6 2*pi])
>> set(gca,'XTickLabel',{'0','pi/2','pi','3pi/2','6','2pi'})
```

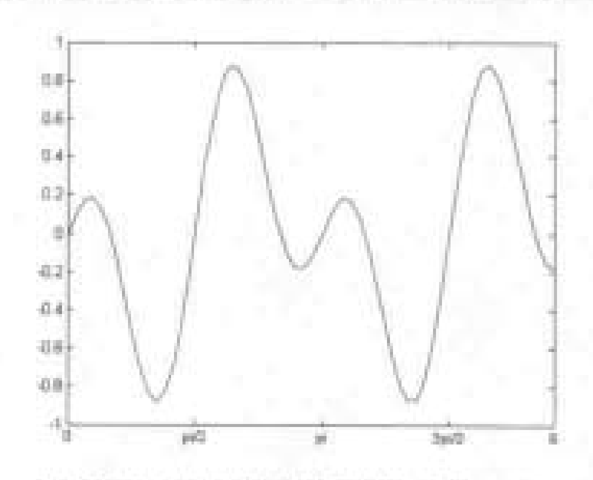


图 23-12　设置坐标轴显示刻度（例 23-10）

### 23.2.7　对数/半对数坐标系绘图

MATLAB 中绘图除了用标准的等比例刻度坐标系，还可以采用对数刻度坐标系。表 23-4 列出了 MATLAB 中和对数/半对数坐标系相关的绘图函数。

表 23-4　对数/半对数坐标系绘图函数

| 函　　数 | 说　　明 |
| --- | --- |
| semilogx | $x$ 轴采用对数刻度的半对数坐标系绘图函数 |
| semilogy | $y$ 轴采用对数刻度的半对数坐标系绘图函数 |
| loglog | $x$ 和 $y$ 轴都采用对数刻度的半对数坐标系绘图函数 |

这三个函数的使用语法和 plot 函数相同，惟一不同的就是绘图结果中的坐标轴。

**例 23-11**　对数/半对数坐标系作图。

**解：**在命令窗口输入：

```
>> x=0:0.1:10;
>> y=exp(-x);
>> subplot(2,2,1)
>> plot(x,y,'r')
>> title('plot')
>> subplot(2,2,2)
>> semilogx(x,y,'--k')
>> title('semilogx')
>> subplot(2,2,3)
>> semilogy(x,y,'-.g','LineWidth',1.5)
>> title('semilogy')
>> subplot(2,2,4)
>> loglog(x,y,':b','LineWidth',0.5')
>> title('loglog')
```

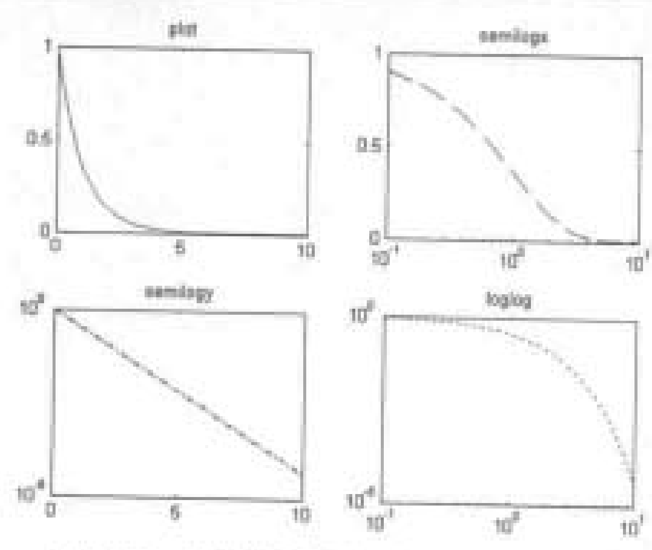



图 23-13　对数/半对数坐标系作图（例 23-11）

## 23.2.8　双纵轴绘图

有些情况下，需要对函数值变化范围差别较大的两组数据同时绘图，如果用叠加绘图模式，则很可能难以从图形中辨识函数值变化范围较小的那组数据变化趋势的细节信息。这时候，一个好的解决方法就是双纵轴绘图。

MATLAB 中提供了 plotyy 函数可以实现双纵轴绘图，对两组数据分别采用左侧纵轴和

右侧纵轴，它们的坐标轴范围各自独立，这样就能从同一幅图中很好地获悉两组数据的变化趋势细节了。

plotyy 函数的基本语法格式是：

plotyy(*X*1,*Y*1,*X*2,*Y*2,'function1','function2')

表示用 function1(*X*1,*Y*1)对第一组数据作图，用 function2(*X*2,*Y*2)对第二组数据作图。当'function2'省略时，则对第二组数据也用 function1(*X*2,*Y*2)作图。当'function1'和'function2'都省略时，则采用默认的 plot 函数对两组数据作图。

另外，为了便于设置左右坐标轴的属性，plotyy 函数还提供了坐标轴句柄返回值，使用格式为：

[AX,H1,H2] = plotyy(…)

其中 AX(1)为左侧坐标轴句柄，AX(2)为右侧坐标轴句柄，H1 为左侧坐标轴下绘制的图线的句柄，H2 为右侧坐标轴下绘制的图线的句柄。利用 set 函数，以这些句柄为输入参数，就可以方便地设置坐标轴和曲线的属性。

**例 23-12** 双纵轴绘图。

**解：** 在命令窗口输入：

```
>> t=0:0.02*pi:7;
>> x=cos(t);
>> y=exp(t);
>> [ax,ha,hb]=plotyy(t,x,t,y)   %句柄都是一些数值，用于标识各个对象
ax =
  153.0322  156.0292
ha =
  155.1194
hb =
  157.0387
>> set(ax(1),'XTick',[0 pi 2*pi],'XTickLabel',{'0','pi','2pi'})
>> set(ha,'LineStyle','--','Color','r')
>> set(ax(2),'XTick',[0 4 7],'XTickLabel',{'0','4','7'})
>> set(ax(2),'YLim',[0 1500],'YTick',[0 exp(4) exp(7) 1500],…
    'YTickLabel',{'0','e^4','e^7','1500'})
```

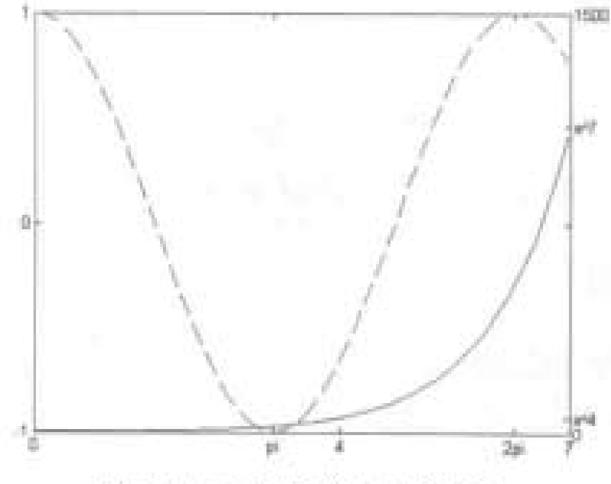



图 23-14 双纵轴绘图（例 23-12）



### 23.2.9　绘图窗口开关控制函数

MATLAB 中大多数图形的属性设置都需要通过 set 函数来完成，但也有少部分图形窗口下的元素可以通过开关函数进行控制。

（1）axis on 和 axis off 可以显示或隐藏当前坐标轴、坐标轴标签和坐标轴刻度。

（2）box on 和 box off 可以显示或者隐藏当前坐标轴的边界线。

（3）grid on 和 gird off 可以显示或者隐藏当前坐标轴下的网格线。

**例 23-13**　开关控制函数（M-File）。

**解**：在命令窗口输入：

```
%Ex23-13 axis/box/grid on-off switch
x=0:0.1:5;
y=10*exp(-x).*x.^2;
subplot(4,2,1)
plot(x,y)
axis on
title('axis on')
subplot(4,2,2)
plot(x,y)
axis off
title('axis off')
subplot(4,2,3)
plot(x,y)
box on
title('box on')
subplot(4,2,4)
plot(x,y)
box off
title('box off')
subplot(2,1,2)
plot(x,y)
grid on
title('grid on')
```

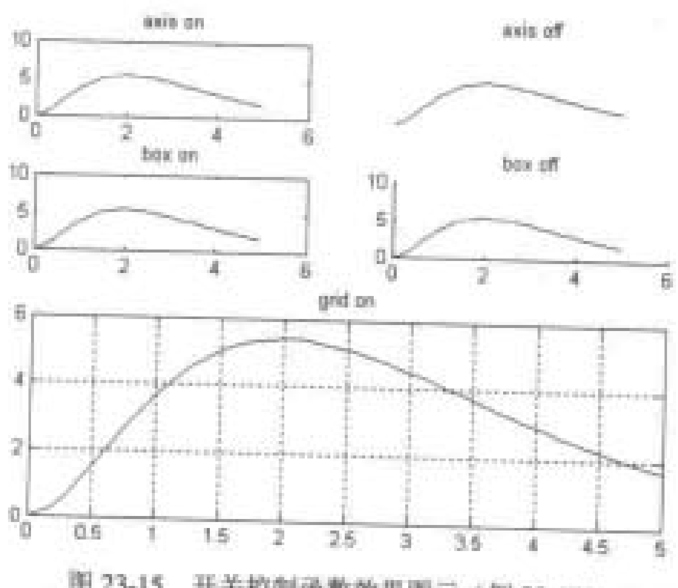



图 23-15　开关控制函数效果图示（例 23-13）

### 23.2.10 设置默认绘图格式循环顺序

在某些单独的项目工程中，用户可能会希望按照自定义的格式依次绘制几条曲线，这时候就可以用 MATLAB 的函数设置用户自定义的绘图格式顺序，这是指在多次绘图中的线型循环顺序和颜色循环顺序。

（1）set(0,'DefaultAxesLineStyleOrder',v)可以将绘图中的线型循环顺序设定为字符串元胞数组 v 指定的值；

（2）set(0,'DefaultAxesColorOrder',v)可以将绘图中的颜色循环顺序设定为字符串元胞数组 v 指定的值。其中 0 是 MATLAB 中根的句柄，因此该设置对所有图形窗口（属于根的子对象）都有效，直到当前 MATLAB 进程中止。

**例 23-14** 设置绘图格式循环顺序。

**解：** 在命令窗口输入：

```
>> set(0,'DefaultAxesLineStyleOrder',{'-x',':o','-.+'})
>> set(0,'DefaultAxesColorOrder',[0 0 0])
>> x=0.5:0.1:1.5;
>> y1=x;
>> y2=x.^2;
>> y3=x.^3;
>> y4=x.^(-1);
>> y5=x.^(-2);
>> y6=x.^(-3);
>> plot(x,y1,x,y2,x,y3,x,y4,x,y5,x,y6)
```

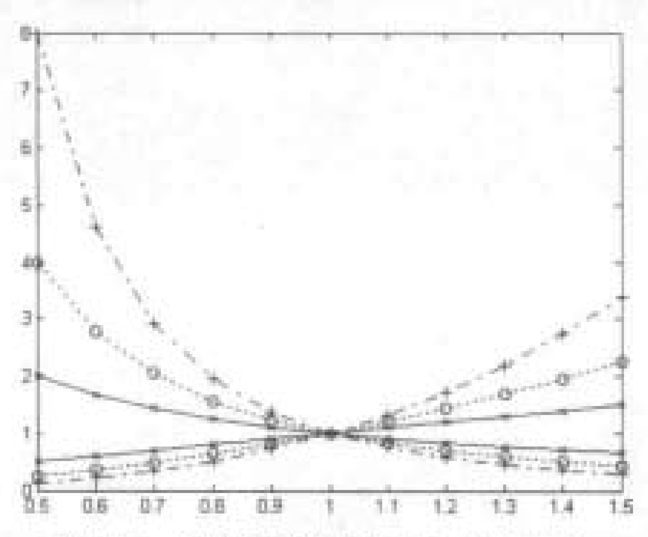



图 23-16 设置绘图格式循环顺序（例 23-14）

需要注意的是，当同时设置了线型循环顺序和颜色循环顺序，且两项设置提供的可选项都大于 1 时，则 MATLAB 实际绘图中以后设置的一项的循环顺序来区分多组图线，而先设置的那项则以设置项中第一个元素为当前设置。

自定义设置了线型循环顺序和/或颜色循环顺序之后，除非重新启动 MATLAB 程序，否则设置持续有效。当然，也可以通过 set 命令清除自定义设置，恢复 MATLAB 的默认设





置。命令是：

（1）set(0,'DefaultAxesLineStyleOrder','remove')

（2）set(0,'DefaultAxesColorOrder','remove')

值得注意的是，MATLAB 默认情况下是通过颜色来区分多组图线的，因此当有效的自定义设置为自定义的颜色循环顺序时，取消自定义设置后，MATLAB 还是会用默认的颜色循环顺序区分多组图线。

要完全理解这里讲述的设置顺序和 MATLAB 的默认方式，读者最好自己书写代码进行相应的测试学习。

### 23.2.11　复数绘图

本节最后简单介绍一下以复数数据为参数的基本绘图。

函数 plot(Z)，当 Z 是复数数组或复数向量时，plot(Z)相当于 plot(real(Z),imag(Z))，实际上是在直角坐标系下将 Z 的各列对应的各点画出并顺次连线。

**例 23-15**　复数绘图。

**解**：在命令窗口输入：

```
>> x=rand(1,5);
>> y=rand(1,5);
>> z=x+y*i
z =
  Columns 1 through 3
   0.6154 + 0.4057i   0.7919 + 0.9355i   0.9218 + 0.9169i
  Columns 4 through 5
   0.7382 + 0.4103i   0.1763 + 0.8936i
>> subplot(2,1,1)
>> plot(z)
>> title('plot(z)')
>> subplot(2,1,2)
>> plot(real(z),imag(z),'-o')
>> title('plot(real(z),imag(z),''-o'')')
```

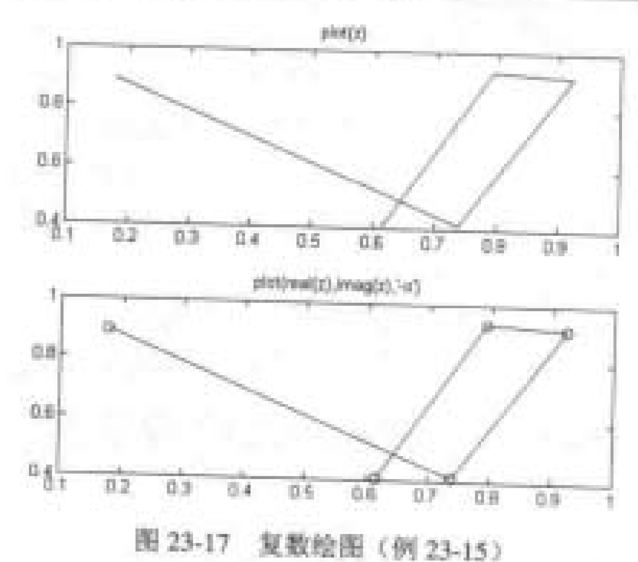


图 23-17　复数绘图（例 23-15）

需要特别说明的是，当对多组复数数据进行绘图时，就没有什么简捷的办法了，必须把每一组复数数据的实部和虚部都分离出来，然后用 plot 函数作图。

## 23.3 图形标注

尽管 MATLAB 绘图中可以通过设置线型和颜色区分多组图线，但图形标注仍然是最广泛使用的提高图形区分度和可读性的方法。MATLAB 中提供了大量的图形标注方法，本节介绍标注函数和工具。

### 23.3.1 图形标注概述

要进行图形标注，首先必须使待编辑的图形处于图形编辑模式打开的状态，这可以通过单击图形窗口的工具菜单（Tools）下的编辑图形子菜单（Edit plot），或者单击图形工具条中的图形编辑模式开关按钮来实现。

当目标图形的编辑模式处于打开状态时，用户就可以对目标图形进行多种多样的标注了。MATLAB 中图形标注的方式可以分为五种：命令窗口中用标注函数标注、通过图形编辑工具条标注、通过插入菜单（Insert）项标注、利用图形面板对象标注和在属性编辑器界面下标注。

MATLAB 中的图形标注函数如表 23-5 所示。大部分图形标注元素的常用属性可以在创建时设定，当然也可以在创建后通过 set 函数设定所有图形标注元素的属性。

表 23-5 图形标注函数

| 函　数 | 说　明 |
|---|---|
| title | 设置标题 |
| xlabel,ylabel | 设置横、纵坐标轴标签 |
| legend | 设置图例 |
| colorbar | 设置颜色条 |
| annotation | 添加文本、线条、箭头、图框等标注元素 |

尽管所有的图形标注都可以通过标注函数实现，但比起编写复杂的标注代码，在图形界面下交互式的标注方式则是更方便和高效的。如果必须用 MATLAB 代码进行标注，也可以用交互式标注结果产生 M-文件（如本章例 23-1 所示），然后对标注的相应部分进行细节调整，这样的方法是容易学习和使用的。

图形编辑工具条在 MATLAB 的默认图形窗口下是不显示的，需要通过单击视图菜单（View）下的图形编辑工具条菜单（Plot Edit Toolbar）来显示，该工具条如图 23-18 所示。

图 23-18 图形编辑工具条





该工具条中各按钮从左向右依次是：填充色、边框色、文字颜色、字体、加粗、斜体、左对齐、居中对齐、右对齐、线条、单箭头、双箭头、带文字标注的箭头、文本、矩形、椭圆、锚定、对齐和分布。它们被五条分割线分割为六组，其中前四组用来设置标注元素的颜色、字体、文字对齐属性，第五组用来添加各种标注元素，最后一组属于特殊用途，本节最后将会讲解。

通过图形编辑工具条只能添加部分 MATLAB 中的图形标注元素，而通过图形窗口的插入菜单（Insert）则可以添加任何 MATLAB 提供的图形标注元素。

MATLAB 图形窗口的插入菜单如图 23-19 所示。

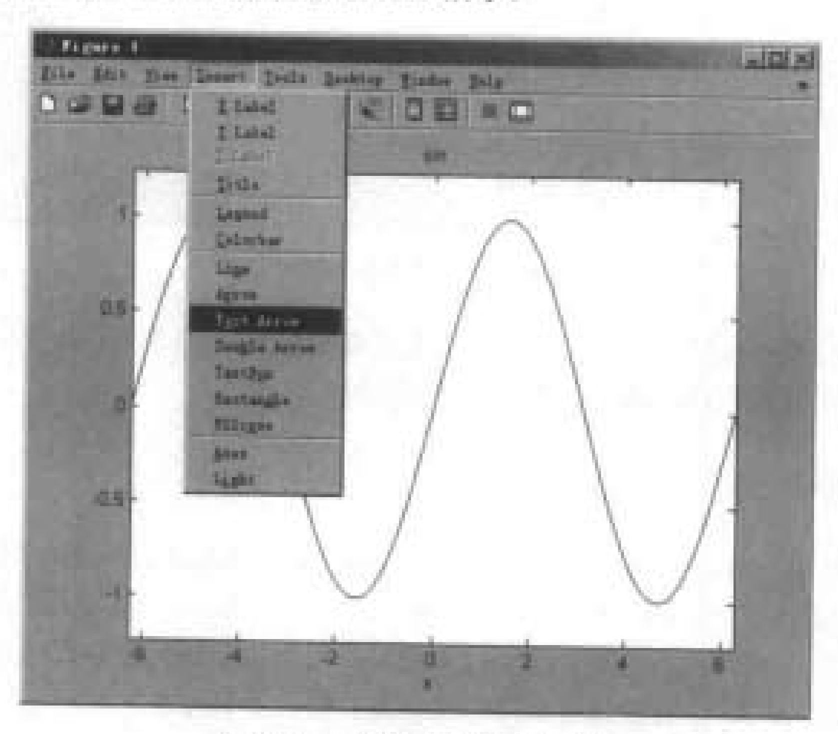


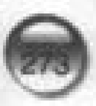
图 23-19 图形窗口的插入菜单项

从插入菜单项可以看出，MATLAB 提供的标注元素包括：坐标轴标签（X Label、Y Label、Z Label）、图形标题（Title）、图例（Legend）、颜色条（Colorbar）、线（Line）、箭头（Arrow）、带文本的箭头（Text Arrow）、双箭头（Double Arrow）、文本框（TextBox）、矩形框（Rectangle）、椭圆框（Ellipse）、坐标轴（Axes）和光影（Light）。其中 Z 轴标签 Zlabel 和光影 Light 只用于三维图形标注中；坐标轴 Axes 是用于在已有图形中添加新的坐标轴，通常不用于标注。

另一个常用的图形界面下的交互标注方法是利用图形面板对象，打开图形面板的方法是单击视图菜单下的图形面板菜单（Figure Palette），效果如图 23-20 所示。

图 23-20 中左侧部分就是图形面板，其最下方是线条和简单图框标注元素。

通过工具条和面板都不能标注图形标题和坐标轴标签，这可以在图形属性编辑器界面下完成，单击视图菜单的属性编辑器子菜单（Property Editor）或者单击图形工具条中的显示绘图工具按钮都可以打开属性编辑器。



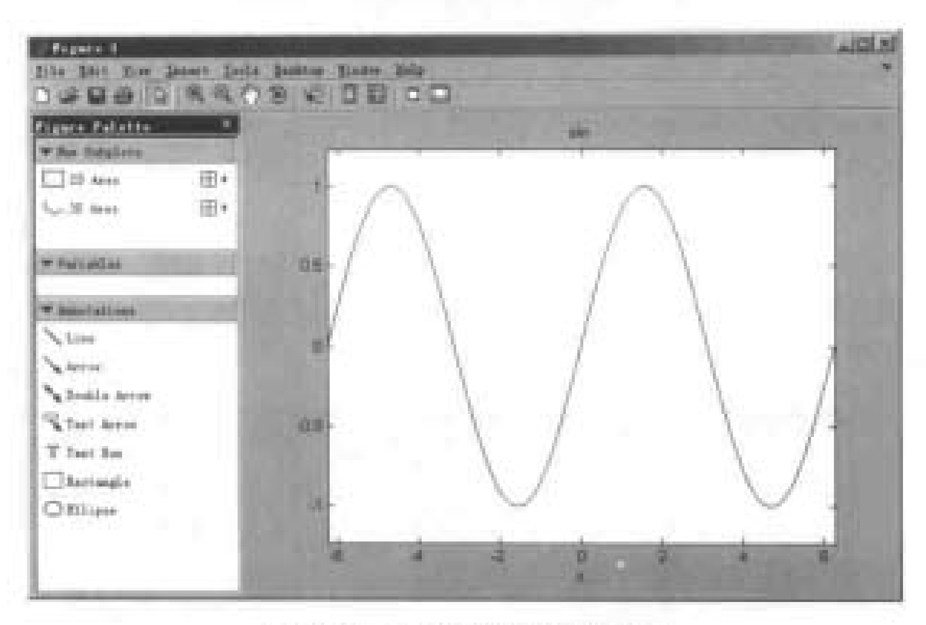


图 23-20　带图形面板的图形窗口

关于图形面板和属性编辑器的应用方法请读者参考本章 23.5 节。本节后续部分重点讲解通过函数、工具条和插入菜单对图形进行标注的方法。

## 23.3.2 图形标题

给图形添加标题可以通过单击插入菜单，选择标题（Title）项或者使用 title 函数。

使用插入菜单的方法时，MATLAB 自动打开图形编辑模式，并在图形顶部出现文字光标，此时可以输入图形标题内容。需要注意的是，图形标题虽然是字符串文本，但不同于一般的文本标注（text），它被固定在图形顶端并且默认居中对齐。

title 函数的常用语法格式如下：

（1）title('string')设置当前绘图区的标题为字符串 string 的值；

（2）title(…,'PropertyName',PropertyValue,…)则可以在添加或设置标题的同时，设置标题的属性，如字体、颜色、加粗等。

**例 23-16**　图形标题函数 title。

**解**：在命令窗口输入：

```
>> title('\ite^{0.2x}sin(x)','FontWeight','Bold')
>> x=0:0.1:5;
>> y=exp(-0.2*x).*sin(x);
>> plot(x,y)
>> title('\ite^{0.2x}sin(x)','FontWeight','Bold')
```

例 23-16 中添加图形标题的同时，设置标题文字为加粗格式，另外，设置文字为斜体（通过 $T_EX$ 语法设置文字为斜体显示，\it 是 $T_EX$ 标记语法格式，表示设置后续文字为斜体），并且^符号后紧跟的大括号内的部分会以上标格式显示。

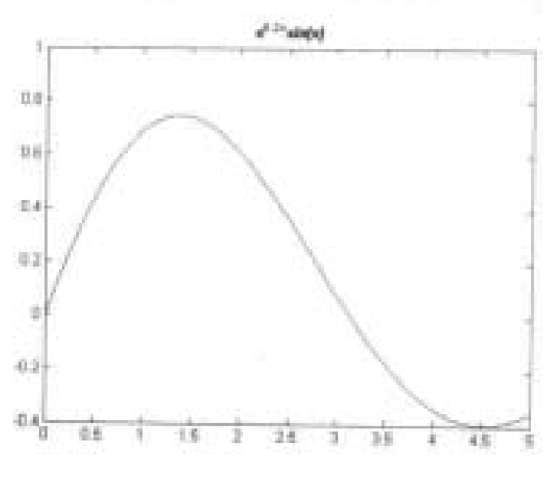


图 23-21　设置图形标题（例 23-16）

关于 MATLAB 中文字的 TEX 标记格式方面的内容，将在 23.3.5 节一一讲述。

### 23.3.3　坐标轴标签

通过插入菜单的 X Label，Y Label 项可以设置图形的横轴和纵轴的标签，默认情况下，横轴标签被安排在横轴下方中间位置并且水平排列，纵轴标签被安排在纵轴左方位置并且垂直排列。坐标轴标签和标题类似，属于文本，但又不同于普通的文本标注。当用户平移、缩放坐标轴时，坐标轴标签会随之变化以适应变化后的坐标轴位置。

添加并设置坐标轴标签的函数是 xlabel 和 ylabel，其常用语法格式为：

（1）xlabel('string')设置横轴标签为字符串值；

（2）xlabel(…,'PropertyName',PropertyValue,…)在设置横轴标签值的同时设置其相关属性，比如文字颜色、旋转角度、字体、加粗等。

**例 23-17**　坐标轴标签。

**解**：在命令窗口输入：

```
>> x=[1990:2:2000];
>> y=[1.25 0.81 2.16 2.73 0.06 0.55];
>> xin=1990:0.2:2000;
>> yin=spline(x,y,xin);
>> plot(x,y,'ob',xin,yin,'-.r')
>> title('1990 年到 2000 年某地区年平均降水量图')
>> xlabel('\it 年份','FontSize',15)
>> ylabel('降雨量','FontSize',8)
```

例 23-17 中设置了横轴的标签为斜体、文字大小为 15；纵轴标签文字大小为 8。如图 23-22，默认情况下纵轴标签垂直显示。实际上可以通过 ylabel('string','Rotation',value)的方法设置纵轴标签的显示方向，value 取值为 0 时标签水平显示；value 取值为 90 时标签从下到上垂直显示；value 为-90 时标签从上到下垂直显示。

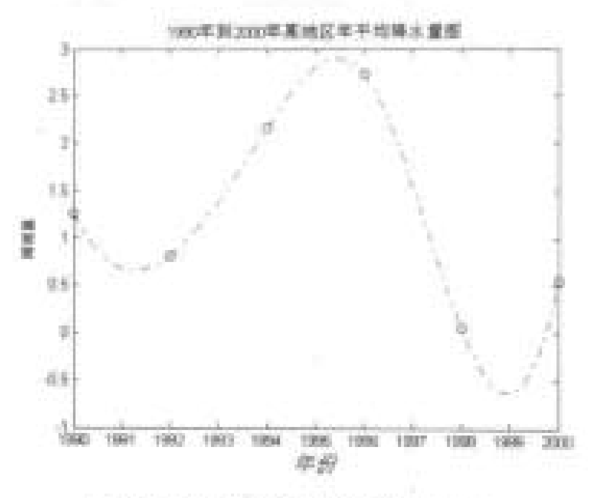


图 23-22 坐标轴标签（例 23-17）

### 23.3.4 图例和颜色条

图例可以用来标注图形中不同颜色、线型的数据组的实际意义。用户可以通过单击插入菜单的图例（Legend）项，或者单击图形工具条的图例按钮，或者通过 legend 命令来添加图例以标注图形中的多组数据。

通过菜单或工具按钮的方法添加图例后，图例的各项文字被设置为 data1、data2 等。要达到用户自定义的设置，使用 lengend 函数是比较方便的。

legend 函数的常用方法为：

（1）legend('string1','string2',…)添加图例，并设置各组数据的图例文字为对应位置字符串值；

（2）legend('off')清除图例；

（3）legend('hide')隐藏图例；

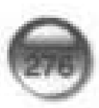
（4）legend('show')显示图例。

**例 23-18** 图例。

**解：**在命令窗口输入：

```
>> x=0:0.02*pi:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> y3=sin(3*x).*cos(x);
>> plot(x,[y1;y2;y3])
>> axis([0 2*pi -1.5 1.5])
>> set(gca,'XTick',[0 pi 2*pi],'XTickLabel',{'0','pi','2pi'})
>> legend('sin(x)','cos(x)','sin(3x)cos(x)')
```

颜色条是用于显示图形中颜色和数值对应关系的，它主要用在三维图形或其二维等高线图形中。用户可以通过单击插入菜单的颜色条（Colorbar）项，或者单击图形工具条的颜色条按钮，或者通过 colorbar 命令来添加颜色条。



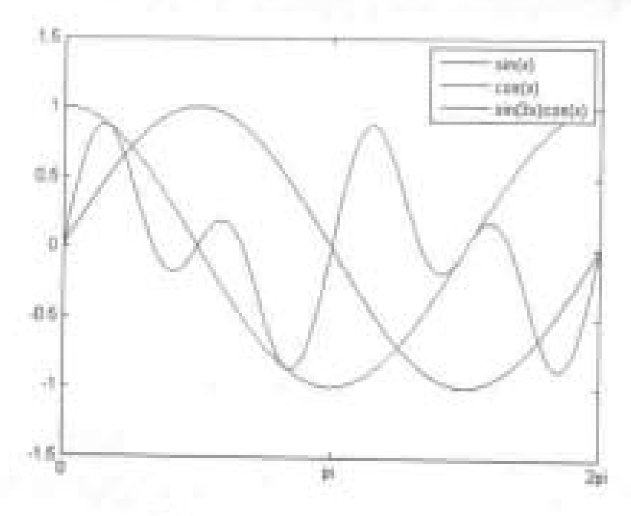



图 23-23 图例（例 23-18）

**例 23-19** 颜色条（注：参考本章 23.4.5）。

**解：** 在命令窗口输入：

```
>> z=peaks(20);
>> [c,h]=contour(z,4);
>> clabel(c,h)
>> colorbar
```

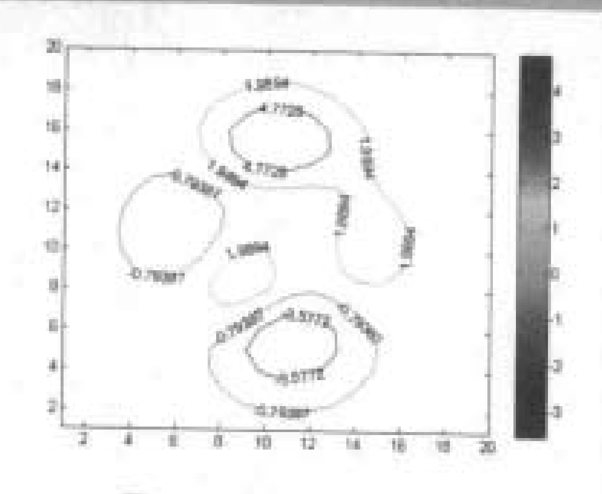


图 23-24 颜色条（例 23-19）

对照图形中的颜色、数值和颜色条中对应颜色、数值，就可以发现颜色条是标志图形中与颜色对应的数值范围了。

### 23.3.5 文本框标注

文本框可以标注在图形中的任何位置，包括坐标轴外的位置。添加文本框可以通过单击插入菜单的文本框（Textbox）项，或者单击图形编辑工具条中的文本框按钮，还可以通过 text 和 gtext 命令来添加文本框标注。

text 和 gtext 命令创建的文本框标注是锚定在图形中的固定位置的，即当坐标轴平移或缩放时，这些文本框标注随同坐标轴一起移动；而通过菜单和工具按钮创建的文本框标注，默认是不锚定在图形上的，即当坐标轴平移或缩放时，这些文本框的显示位置不动，除非用户自定义设置了文本框锚定。

text 是纯命令行文本框标注函数，其常用语法格式为：

（1）text(x,y,'string')，在(x,y)坐标点的位置进行文本框标注，文本内容为 string 值；

（2）text(…'PropertyName',PropertyValue…)，可以在文本框标注的同时设置其格式，这主要是设置文本字号和对齐方式。

gtext 是交互式文本框标注函数，其常用语法格式为：

（1）gtext('string')可以在鼠标点击的位置标注一个单行文本框；

（2）gtext({'string1','string2','string3',…})可以在鼠标点击位置标注一个多行文本框；

（3）gtext({'string1';'string2';'string3';…})可以在通过多次鼠标点击标注多个文本框。

**例 23-20** 文本框标注。

**解**：在命令窗口输入：

```
>> x=0:0.02*pi:2*pi;
>> y=sin(x)+cos(x);
>> plot(x,y,pi,sin(pi)+cos(pi),'o')
>> text(pi,sin(pi)+cos(pi),['sin(\pi)+cos(\pi)=',num2str(sin(pi)+ cos (pi))])
>> gtext({'gtext1-line1','gtext1-line2'})
>> gtext({'gtext2-1';'gtext2-2'})
```

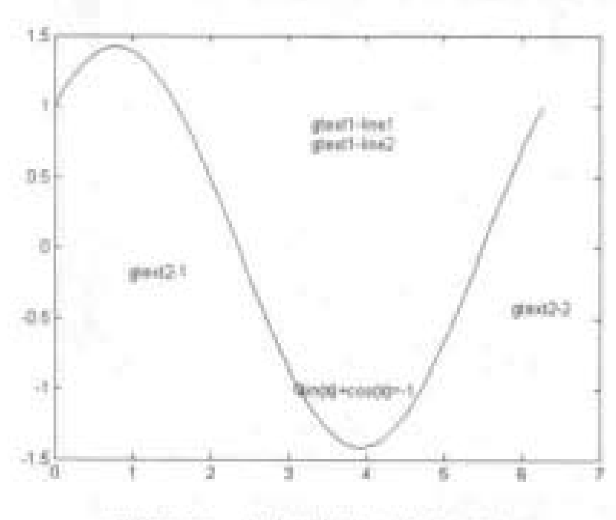



图 23-25 文本框标注（例 23-20）

例 23-20 中，\pi 是 $\mathrm{T_EX}$ 标记语言，通过 $\mathrm{T_EX}$ 标记语言可以设置多种常用符号，如希腊字母、数学符号、箭头等，MATLAB 中可用的 $\mathrm{T_EX}$ 字符序列和对应符号如图 23-26 所示。

通过 $\mathrm{T_EX}$ 标记语言，还可以设置字体、颜色等，如表 23-6 所示。

| TEX字符序列 | 符号 | TEX字符序列 | 符号 | TEX字符序列 | 符号 |
|---|---|---|---|---|---|
| \alpha | α | \upsilon | υ | \sim | ~ |
| \beta | β | \phi | ϕ | \leq | ≤ |
| \gamma | γ | \chi | χ | \infty | ∞ |
| \delta | δ | \psi | ψ | \clubsuit | ♣ |
| \epsilon | ϵ | \omega | ω | \diamondsuit | ♦ |
| \zeta | ζ | \Gamma | Γ | \heartsuit | ♥ |
| \eta | η | \Delta | Δ | \spadesuit | ♠ |
| \theta | θ | \Theta | Θ | \leftrightarrow | ↔ |
| \vartheta | ϑ | \Lambda | Λ | \leftarrow | ← |
| \iota | ι | \Xi | Ξ | \uparrow | ↑ |
| \kappa | κ | \Pi | Π | \rightarrow | → |
| \lambda | λ | \Sigma | Σ | \downarrow | ↓ |
| \mu | µ | \Upsilon | ϒ | \circ | º |
| \nu | ν | \Phi | Φ | \pm | ± |
| \xi | ξ | \Psi | Ψ | \geq | ≥ |
| \pi | π | \Omega | Ω | \propto | ∝ |
| \rho | ρ | \forall | ∀ | \partial | ∂ |
| \sigma | σ | \exists | ∃ | \bullet | • |
| \varsigma | ς | \ni | ∍ | \div | ÷ |
| \tau | τ | \cong | ≅ | \neq | ≠ |
| \equiv | ≡ | \approx | ≈ | \aleph | ℵ |
| \Im | ℑ | \Re | ℜ | \wp | ℘ |
| \otimes | ⊗ | \oplus | ⊕ | \oslash | ∅ |
| \cap | ∩ | \cup | ∪ | \supseteq | ⊇ |
| \supset | ⊃ | \subseteq | ⊆ | \subset | ⊂ |
| \int | ∫ | \in | ∈ | \o | ο |
| \rfloor | ⌋ | \lceil | ⌈ | \nabla | ∇ |
| \lfloor | ⌊ | \cdot | · | \ldots | ... |
| \perp | ⊥ | \neg | ¬ | \prime | ′ |
| \wedge | ∧ | \times | x | \0 | ∅ |
| \rceil | ⌉ | \surd | √ | \mid | ∣ |
| \vee | ∨ | \varpi | ϖ | \copyright | © |
| \langle | 〈 | \rangle | 〉 | | |

图 23-26　$\mathrm{T_EX}$ 字符序列和对应符号

**表 23-6　设置字体、颜色的 $\mathrm{T_EX}$ 标记序列**

| $\mathrm{T_EX}$ 标记序列 | 功　能 |
|---|---|
| \bf | 字体加粗 |
| \it | 字体倾斜 |
| \rm | 正常字体 |
| \fontname{fontname} | 采用指定字体 |
| \fontsize{fontsize} | 指定字号 |
| \color{colorname} | 指定颜色<br>colorname 可以指定八种基本颜色 red, green, yellow, magenta, blue, black, white 和四种 Simulink 颜色 gray, darkGreen, orange,和 lightBlue<br>注：colorname 不能采用单字符缩写 |

读者可能已经注意到，所有 $\mathrm{T_EX}$ 标记序列都用反斜杠（\）引导。另外，使用表 23-6 中的设置时，经常和花括号配合使用，花括号内的 $\mathrm{T_EX}$ 标记设置只在花括号内有效。



TEX 标记序列可以用在所有支持 TEX 解释的文本型标注中，如标题、坐标轴标签、文本框文字等。

例 23-21　利用 TEX 标记序列进行文本标注。

解：在命令窗口输入：

```
>> figure         %新建图形窗口
>> axes           %在当前图形窗口新建坐标轴
>> gtext({'符号: \leftrightarrow\pi\Omega';'{\bf 加粗}{\it 斜体}'; …
'\fontsize{20}{\color{red}Red}{\color{blue}Blue}'; …
'\color{magenta}{\fontsize{8}8}{\fontsize{18}18}'})
```

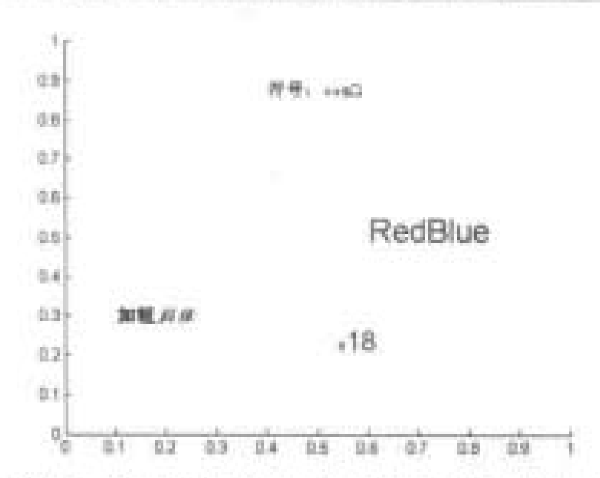



图 23-27　利用 TEX 标记序列进行文本标注（例 23-21）

前面已经提到 text 命令，可以在文本框标注的同时设置其对齐格式，采用的语法格式为：text(…'PropertyName',PropertyValue…)。表 23-7 列出 MATLAB 中提供的文本框的对齐属性和可以选择的对齐方式。

表 23-7　文本框对齐属性和对齐方式

| 对齐属性 PropertyName | 对齐方式 PropertyValue | 说明 |
|---|---|---|
| HorizontalAlignment | Left（MATLAB 默认） | 水平方向左对齐 |
| | Center | 水平方向居中对齐 |
| | Right | 水平方向右对齐 |
| VerticalAlignment | Middle（MATLAB 默认） | 垂直方向居中对齐 |
| | Top | 垂直方向顶端对齐 |
| | Cap | 垂直方向加帽对齐 |
| | Baseline | 垂直方向基线对齐 |
| | Bottom | 垂直方向底端对齐 |

例 23-22 通过 MATLAB 子图绘制比较表 23-7 所列的各种对齐方式。

例 23-22　文本框对齐方式（M-File）。

解：在命令窗口输入：

```
subplot(2,3,1)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Left','FontSize',15,'HorizontalAlignment','Left', …
```





```
    'LineStyle','-.','EdgeColor','r')    %设置文本框边界线型和颜色
title('水平对齐-Left')
subplot(2,3,2)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Center','FontSize',15,'HorizontalAlignment','Center', ...
    'LineStyle','-.','EdgeColor','r')
title('水平对齐-Center')
subplot(2,3,3)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Right','FontSize',15,'HorizontalAlignment','Right', ...
    'LineStyle','-.','EdgeColor','r')
title('水平对齐-Right')
subplot(4,3,7)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Middle','FontSize',15,'VerticalAlignment','Middle', ...
    'LineStyle','-.','EdgeColor','r')
title('垂直对齐-Middle')
subplot(4,3,8)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Top','FontSize',15,'VerticalAlignment','Top', ...
    'LineStyle','-.','EdgeColor','r')
title('垂直对齐-Top')
subplot(4,3,9)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Cap','FontSize',15,'VerticalAlignment','Cap', ...
    'LineStyle','-.','EdgeColor','r')
title('垂直对齐-Cap')
subplot(4,2,7)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Baseline','FontSize',15,'VerticalAlignment','Baseline', ...
    'LineStyle','-.','EdgeColor','r')
title('垂直对齐-Bottom')
title('垂直对齐-Baseline')
subplot(4,2,8)
plot(1,1,'+','MarkerSize',30)
text(1,1,'Bottom','FontSize',15,'VerticalAlignment','Bottom', ...
    'LineStyle','-.','EdgeColor','r')
title('垂直对齐-Bottom')
```

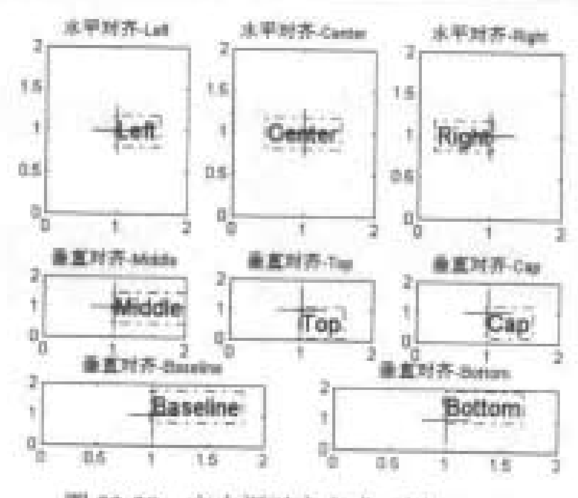



图 23-28　文本框对齐方式（例 23-22）

最后要稍作强调的是，text 和 gtext 也都可以标注多行文本，还可以标注数值转换得到的字符串，例 23-20 和例 23-21 中都已示范过，此处不再赘述。

### 23.3.6 数据点标记

MATLAB 中还可以对特定的数据进行数据点标记，只需要在图形窗口中单击图形工具条中的数据点标记按钮，就可以完成此项操作。图 23-29 就是一个典型的带有数据点标记的图形窗口。

通过数据点标记按钮不但可以标记绘图中采用的数据点，还可以标记曲线上的插值点。在进行数据点标记时，单击鼠标右键，在弹出的快捷菜单中可以看到有数据点选择方法、显示样式、新建数据点标记、删除当前数据点标记、删除所有数据点标记和将标记处数值导入到 MATLAB 工作区等子菜单项。

数据点选择方法有两种：

（1）跟随鼠标位置，需要采用插值方法得到当前数据点处的坐标；

（2）指向最近的绘图原始数据点，只标记绘图用数据点的坐标。

显示也有两种样式：子窗口模式和标贴模式。

图 23-29 中在标记 $X$=0.5108 的数据点时采用了非默认的鼠标跟随的数据点选择方法，显示采用了默认的标贴模式。

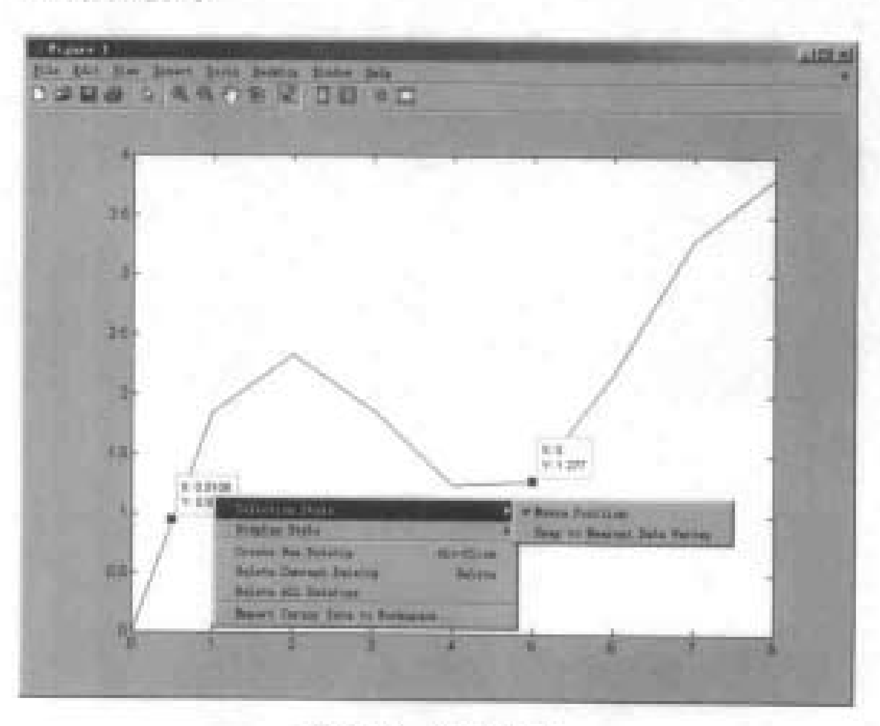


图 23-29 数据点标记

### 23.3.7 箭头和图框标注

在图形中添加箭头标注主要是起到明确的指示作用，而图框标注则能提高图形的美观



程度。这两类元素的标注可以通过 annotation 函数来实现。

实际上 annotation 可以对图形进行任意类型的对象标注，其标注箭头和图框的语法为：

（1）annotation('line',x,y)在指定的坐标位置标注线段；

（2）annotation('arrow',x,y)在指定的坐标位置标注箭头；

（3）annotation('doublearrow',x,y)在指定的坐标位置标注双箭头；

（4）annotation('textarrow',x,y)在指定的坐标位置标注带文字的箭头；

（5）annotation('ellipse',[x y w h])在指定的位置标注椭圆框；

（6）annotation('rectangle',[x y w h])在指定的位置标注矩形框；

（7）annotation(…,'PropertyName',PropertyValue,…)在标注的同时设置标注对象的属性。

实际上，更简捷方便的标注箭头和图框的方法是通过插入菜单、图形编辑工具条按钮。这时候只需要单击相应的菜单项或工具按钮，然后用鼠标交互式地指定标注位置即可，而不需要向 annotation 函数那样进行复杂的坐标位置的设定了。

交互式标注的箭头和图框也可以编辑各种属性，只需要在图形编辑模式打开的情况下，双击标注对象，就会在窗口底端出现属性编辑器界面，其中可以进行多种属性设置；另外，简单的属性设置可以在图形编辑工具条中进行，比如边界线条颜色、填充色等。

图 23-30 显示的是用交互式方法标注箭头和图框，并用属性编辑器设置标注对象各种属性之后的图形窗口。

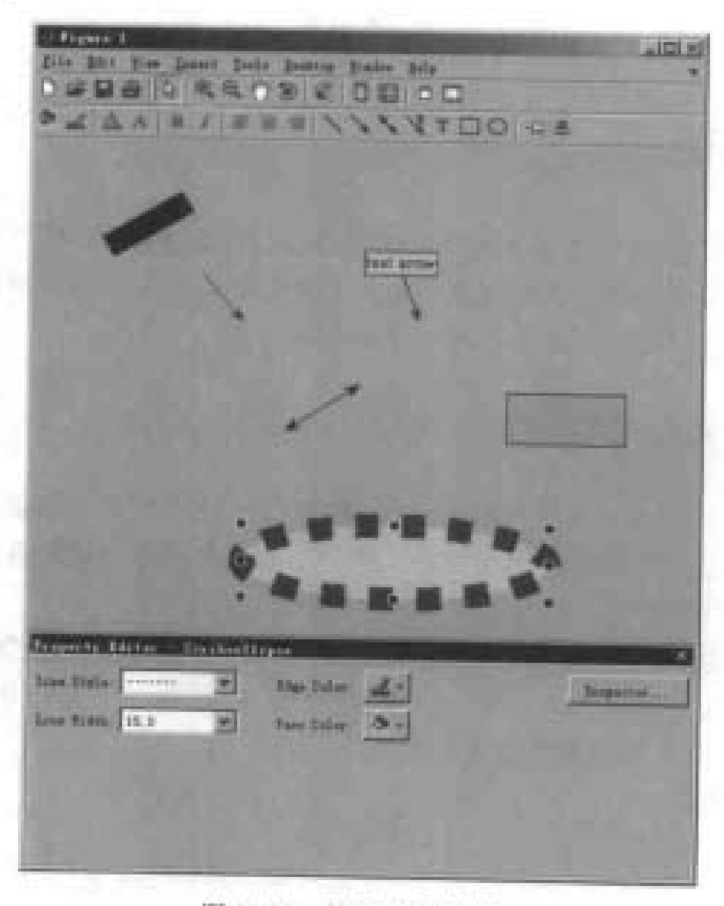
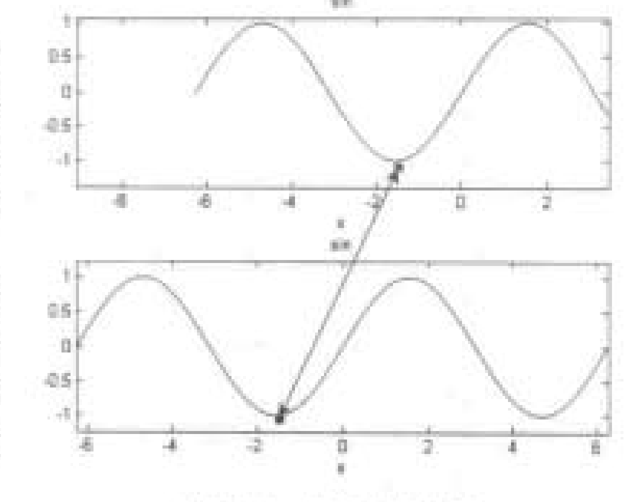


图 23-30 箭头和图框标注

### 23.3.8 锚定图形标注对象


图 23-31 标注对象的锚定

通过菜单和工具按钮添加的标注对象，不管是文本框，还是箭头、图框，显示时都是固定显示在图形窗口的某个位置的。但很多时候标注就是为了达到对局部数据的指示作用，经常需要将某一标注对象相对于坐标轴进行锚定操作。

要将某个标注对象相对于坐标轴进行锚定，只需要单击图形编辑工具条中的锚定按钮，然后指定要锚定的对象的锚定点；或者选择目标对象，单击鼠标右键在弹出的快捷菜单中选择锚定。

对象相对于坐标轴锚定后，对图像进行平移，缩放等操作时，被锚定的对象的坐标位置不变，即会随同坐标轴的变动而变动。如图 23-31 中的双箭头指示了两个子图中同一位置点，第一子图中的箭头点锚定在第一子图的坐标轴上，因此，平移第一子图后，该点位置随着变动以保持坐标位置的不变。

去除对象的锚定也可以在右键菜单中进行设置。

## 23.4 特殊绘图

MATLAB 中除了可以通过 plot 等相关函数绘制二维线条图外，还有许多特殊的绘图指令，通过这些特殊绘图，使用者可以方便地获悉单个数据在整体的数据集中所占的比例，数据点的分布，数据分布的向量信息以及等高线等。

### 23.4.1 柱状图和面积图

MATLAB 中可以用 bar 或者 barh 指令绘制柱状图，它们把单个数据显示为纵向或者横向的柱条，这在察看变量的时间变化趋势、比较不同组数据集、比较各个单独数据点在总体中的比重等方面都有重要的指导意义。

bar 函数可以按照 bar(data,'mode')的语法格式接收模式参数，默认情况下为'grouped'模式，这时候 bar 函数把数组 data 的每一行看作一组，画在同一个水平坐标位置。若指定为'stacked'，则把每一组的数据累叠起来绘图。

**例 23-23** 柱状图。

**解**：在命令窗口输入：

```
>> X=[3 2 1;4 2 7;3 6 9;5 1 7];
>> subplot(1,2,1)
```





```
>> bar(X)
>> subplot(1,2,2)
>> barh(X,'stacked')
```

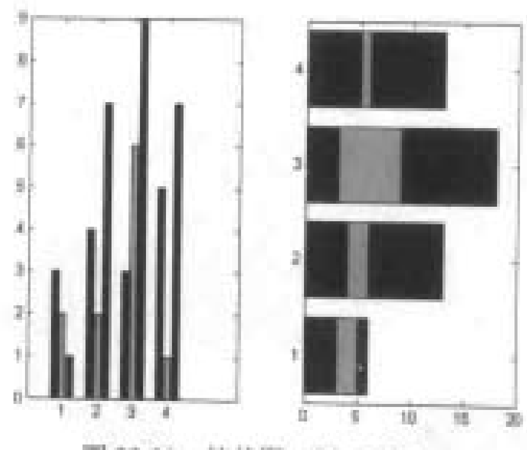


图 23-32　柱状图（例 23-23）

area 函数用来绘制面积图，和累叠模式的柱状图类似，面积图也是把每一组数据点累叠绘制，不过它把每一个数据集合的相邻点用线条连起来，并且把每一个数据集合所在区域用不同的颜色填充。

**例 23-24**　面积图。

**解：**在命令窗口输入：

```
>> part1=[4 6 7 9 2 3]';
>> part2=[2 1 3 7 9 8]';
>> area([part1 part2])
>> gtext('total=part1+part2')
```

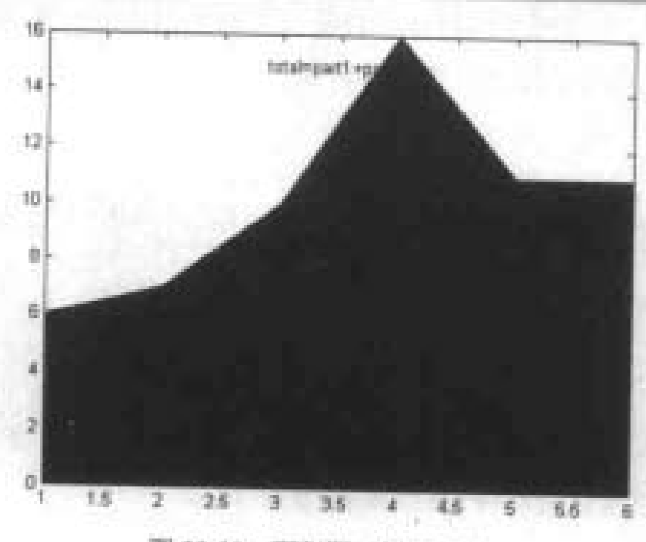


图 23-33　面积图（例 23-24）

### 23.4.2　饼图

饼图可以用来显示每一个元素在总体中的比例，MATLAB 中绘制二维饼图的函数是 pie。

若输入数据总和超过1，pie函数会自动计算每一数据在总体中的比例；而当输入数据总和小于1时，pie只绘制输入数据指定的各部分，不足1的部分空缺处理。

**例23-25** 饼图。

**解**：在命令窗口输入：

```
>> x=rand(1,5)
x =
   0.5028    0.7095    0.4289    0.3046    0.1897
>> y=[0.2 0.45 0.1];
>> subplot(1,2,1)
>> pie(x)
>> subplot(1,2,2)
>> pie(y)
```

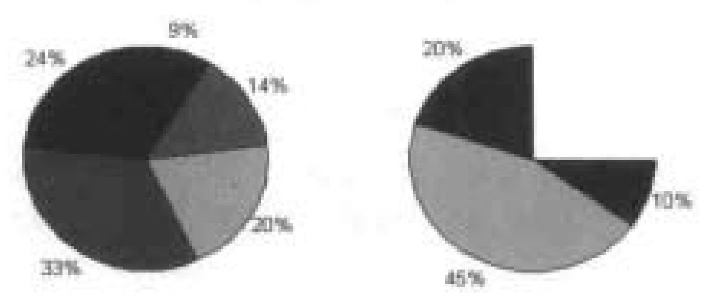



图23-34 饼图（例23-25）

### 23.4.3 直方图

直方图也称为频数直方图，它用来显示已知数据集的分布情况，已知数据集的数据范围被分割成若干个区间，直方图中用每一个柱条代表处于该区间中的数据点数目。MATLAB中通过hist函数来绘制直角坐标下的频数直方图。

**例23-26** 频数直方图。

**解**：在命令窗口输入：

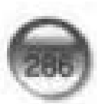
```
>> x=randn(1000,1);
>> y=randn(1000,3);
>> subplot(3,1,1)
>> hist(x)
>> subplot(3,1,2)
>> hist(x,50)
>> subplot(3,1,3)
>> hist(y,25)
```

极坐标下的直方图也称为玫瑰图，绘制函数是rose。

**例23-27** 玫瑰图。

**解**：在命令窗口输入：

```
>> x=rand(1000,1)*1000;
>> theta=x*pi/180;
>> rose(theta)
>> set(findobj(gca,'Type','line'),'LineWidth',2.0)
```



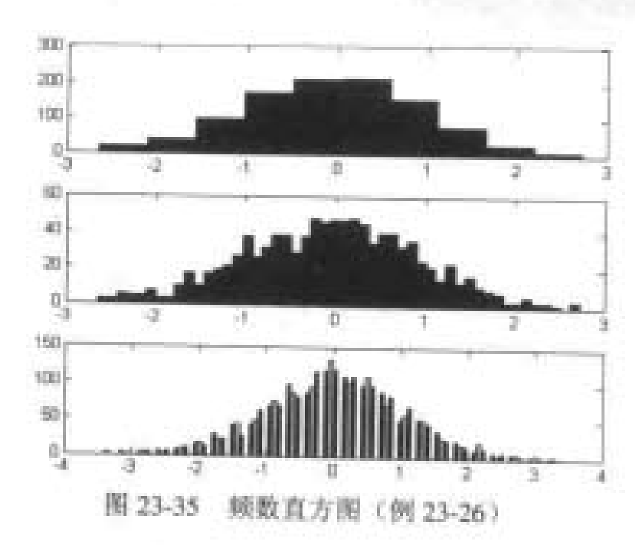


图 23-35　频数直方图（例 23-26）

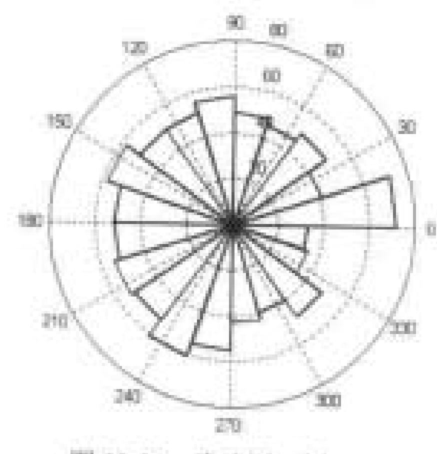


图 23-36　玫瑰图（例 23-27）

### 23.4.4　离散数据绘图

显示离散数据的变化趋势，除了 bar 绘制的柱状图外，还有 stem 绘制的火柴杆图和 stairs 绘制的阶梯图。

火柴杆图是把每一个数据点用一个垂直于横轴的火柴棒来表示，火柴头的位置表示数据点。火柴杆图中可以定制火柴杆的线型、颜色和火柴头的形状、填充等属性。

**例 23-28**　火柴杆图。

**解：**在命令窗口输入：

```
>> t=0:0.2:10;
>> y=exp(-0.2*t).*cos(7*t);
>> subplot(2,1,1)
>> stem(t,y)
>> hold on
>> plot(t,y)
>> plot(t,y,'r')
>> subplot(2,1,2)
>> stem(t,y,'-.dg','fill')
```

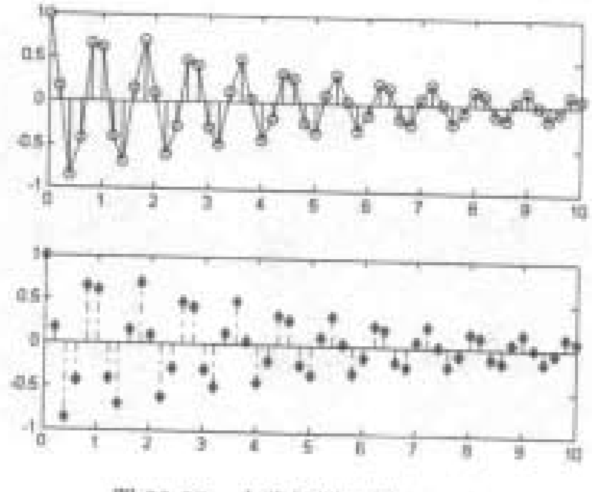


图 23-37　火柴杆图（例 23-28）



**例 23-29** 阶梯图。

**解**：在命令窗口输入：

```
>> t=0:0.5:10;
>> y=exp(-0.2*t).*cos(t);
>> stairs(t,y)
>> hold on
>> plot(t,y,'r')
```

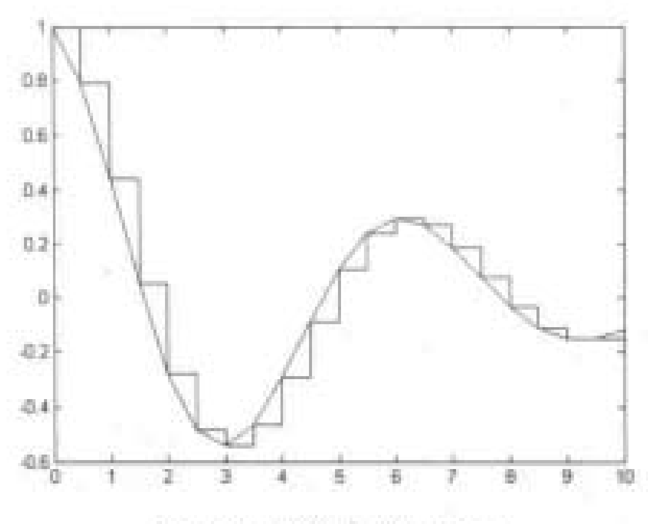



图 23-38 阶梯图（例 23-29）

### 23.4.5 等高线图

等高线图最常用于显示多元函数（尤其是二元函数）的函数值变化趋势。MATLAB 中用 contour 函数绘制一般的等高线图，clabel 可以用来标注等高线图中的函数值，contourf 函数则是绘制填充模式的等高线图。

**例 23-30** 等高线图。

**解**：在命令窗口输入：

```
>> z=peaks;
>> subplot(2,1,1)
>> contour(z)
>> subplot(2,1,2)
>> [c,h]=contour(z,[0.8 1.5]);
>> clabel(c,h)        %如图 23-39 所示
>> figure             %新建图形窗口
>> subplot(2,1,1)
>> [c,h]=contour(z,4);
>> clabel(c,h)
>> subplot(2,1,2)
>> contourf(z,4)            %如图 23-40 所示
```





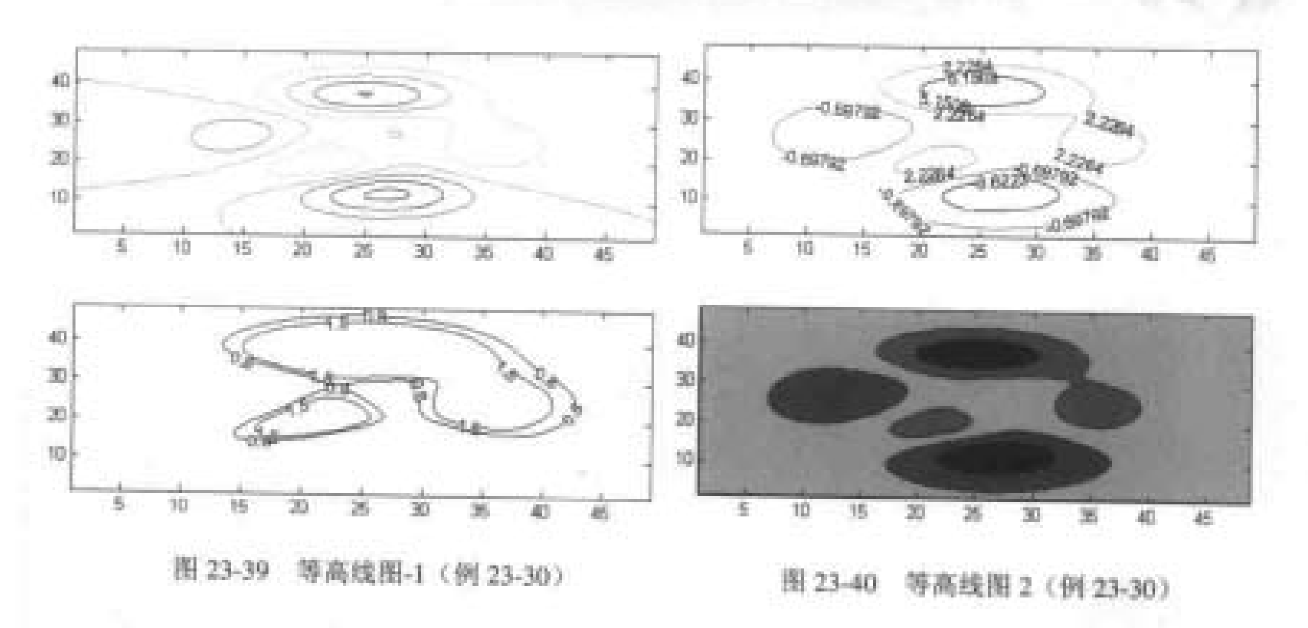



图 23-39　等高线图-1（例 23-30）

图 23-40　等高线图 2（例 23-30）

## 23.4.6　向量图

有些情况下，需要用图形表示数据的方向信息，这时候就需要绘制向量图。MATLAB 中常用的向量图包括罗盘图、羽毛图和向量场图。

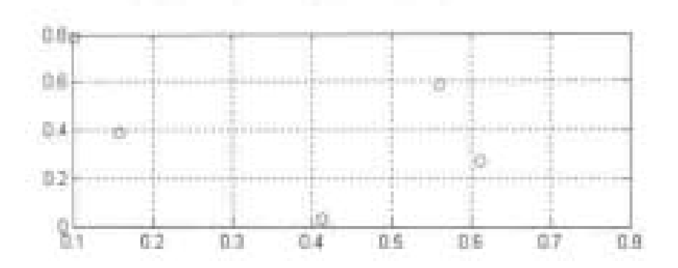
compass 函数可以绘制罗盘图，compass 函数接收直角坐标参数，而在绘制出的罗盘图中，每一个数据点被表示为极坐标下一条从原点出发的带箭头的线段。

**例 23-31**　罗盘图。

**解**：在命令窗口输入：

```
>> x=rand(1,5)
x =
   0.6119    0.1030    0.1583    0.4136    0.5604
>> y=rand(1,5)
y =
   0.2687    0.7843    0.3879    0.0310    0.5855
>> subplot(2,1,1)
>> plot(x,y,'ro')
>> grid on
>> subplot(2,1,2)
>> compass(x,y)
```

函数 feather 用来绘制羽毛图。feather 也接收直角坐标参数，与罗盘图不同的是，羽毛图是在直角坐标系下绘制的，每一个数据点也被表示为带箭头的线段，不过其起点是 $x$ 轴上间隔单位长度的刻度点。



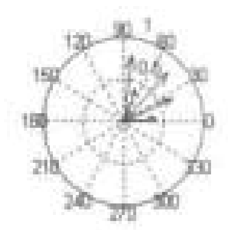


图 23-41　罗盘图（例 23-31）

例 23-32　羽毛图。

**解**：在命令窗口输入：

```
>> x=rand(1,5)
x =
   0.5586    0.2007    0.0874    0.9332    0.2594
>> y=rand(1,5)
y =
   0.2042    0.0492    0.6062    0.5463    0.0958
>> subplot(2,1,1)
>> plot(x,y,'ro')
>> subplot(2,1,2)
>> grid on
>> feather(x,y)
```

向量场图的绘制函数是 quiver，其语法格式为：

quiver($x$,$y$,$u$,$v$)

表示以($x$,$y$)为起点，用箭头表示($u$,$v$)代表的向量。向量场图也是直角坐标系下的向量图，最常用于描绘梯度场。

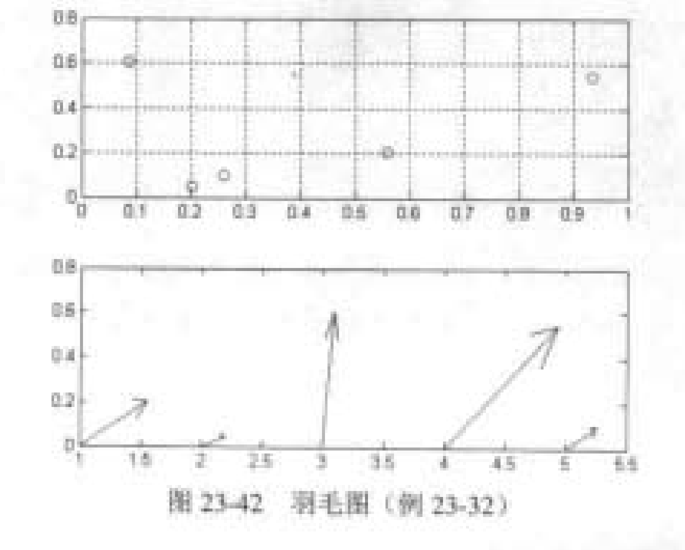


图 23-42　羽毛图（例 23-32）



例 23-33　向量场图。

解：在命令窗口输入：

```
>> [x,y,z]=peaks(20);
>> contour(x,y,z,10)
>> [u,v]=gradient(z);
>> hold on
>> quiver(x,y,u,v)
```

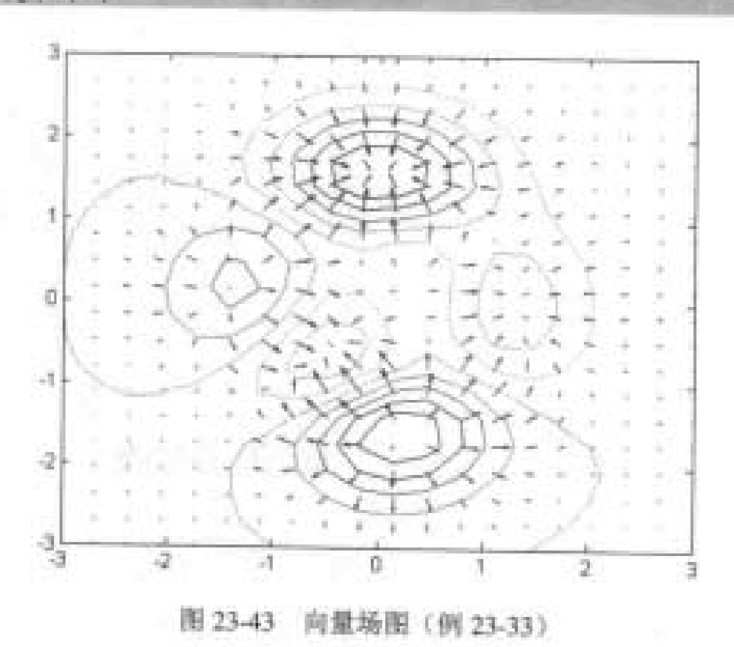


图 23-43　向量场图（例 23-33）

## 23.4.7　其他特殊绘图指令

除了前面介绍的特殊绘图指令，MATLAB 中还有许多其他的特殊绘图函数，表 23-8 仅列出前面没有介绍到，但比较常用的特殊绘图指令。

表 23-8　其他特殊绘图指令

| 函　数 | 说　明 |
| --- | --- |
| comet | 画彗星图 |
| pareto | 画 pareto 图 |
| scatter | 画散点图 |
| fill | 画实心图 |
| polyarea | 画输入数组参数确定的多边形的实心图 |
| polymatrix | 画输入数组的关系图 |

例 23-34　其他特殊绘图指令-1。

解：在命令窗口输入：

```
>> x=rand(1,5)
x =
    0.4152    0.8673    0.6249    0.0552    0.4041
```

```
>> y=rand(1,5)
y =
    0.3020    0.1523    0.3092    0.0033    0.4374
>> subplot(2,2,1)
>> fill(x,y,'k')
>> title('实心图')
>> subplot(2,2,2)
>> pareto(x)
>> title('pareto 图')
>> subplot(2,2,3)
>> scatter(x,y)
>> title('散点图')
>> subplot(2,2,4)
>> comet(x,y)
>> title('彗星图')
```

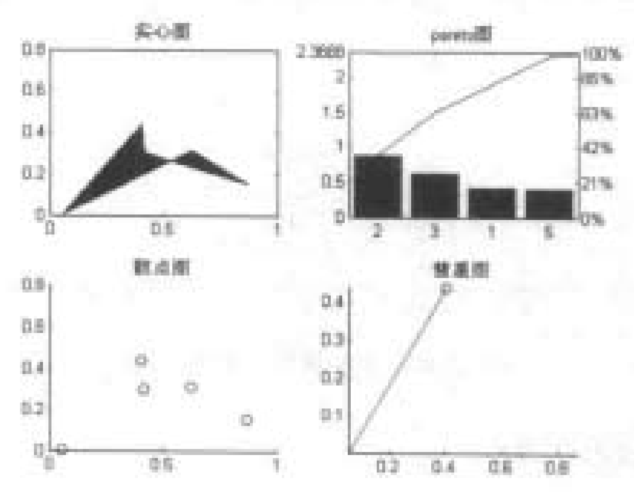
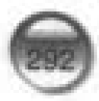



图 23-44　其他特殊绘图指令-1（例 23-34）

例 23-35　其他特殊绘图指令-2。

解：在命令窗口输入：

```
>> subplot(2,1,1)
>> a=[1 2 6 7 9 1];
>> b=[5 1 3 2 4 5];
>> A=polyarea(a,b)
A =
   16.5000
>> fill(a,b,'b')
>> title(['Area=' num2str(A)])
>> subplot(2,1,2)
>> x=[1 2 6;7 9 1];
>> y=[5 1;4 5];
>> plotmatrix(x,y)
>> title('Matrix x vs Matrix y')
```



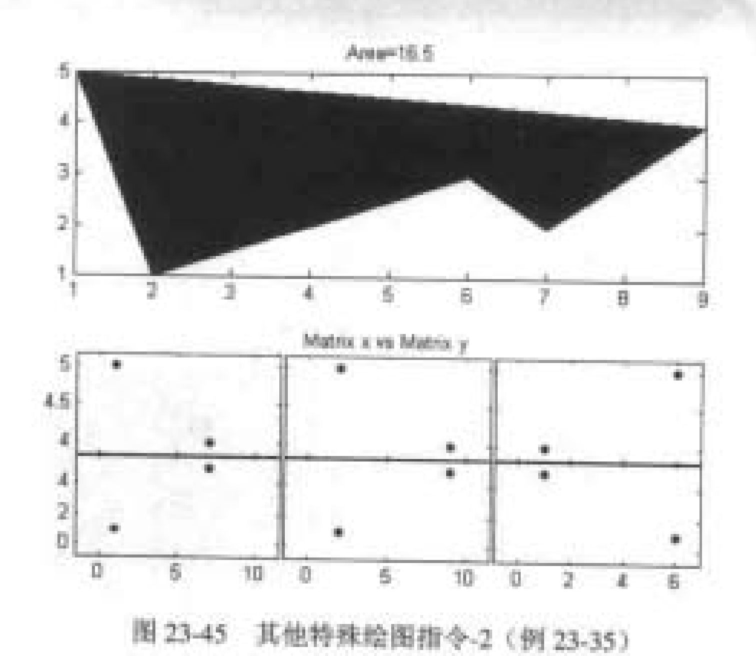



图 23-45　其他特殊绘图指令-2（例 23-35）

### 23.4.8　函数绘图

MATLAB 中二维的函数绘图指令如表 23-9 所示。

表 23-9　二维函数绘图

| 函　数 | 说　明 |
|---|---|
| fplot(fun,limits) | 在 limits 指定的坐标范围内绘制字符串或函数句柄 fun 指定的函数图形 |
| ezplot(fun,[xmin,xmax,ymin,ymax]) | 在指定的坐标范围内绘制字符串或函数句柄 fun 指定的函数图形 |
| ezpolar(fun,[*a*,*b*]) | 在指定的弧度范围内绘制字符串或函数句柄 fun 指定的函数极坐标图 |
| ezcontour(fun) | 绘制字符串或函数句柄 fun 指定的函数的等高线图 |
| ezcontourf(fun) | 绘制字符串或函数句柄 fun 指定的函数的等高线填充图 |

**例 23-36**　函数绘图。

**解：** 在命令窗口输入：

```
>> subplot(2,2,1)
>> fplot(@sin,[0 2*pi])
>> title('y=sin(x)')
>> subplot(2,2,2)
>> f='cos(x)+sin(3*x)';
>> ezplot(f,[-2*pi 2*pi -3 3])
>> title('y=cos(x)+sin(3x)')
>> subplot(2,2,3)
>> ezpolar('2+3*sin(t)+5*cos(t)',[0 2*pi])
>> title('\rho=sin(\theta)+cos(\theta)')
>> subplot(2,2,4)
>> ezcontourf(@peaks)
```

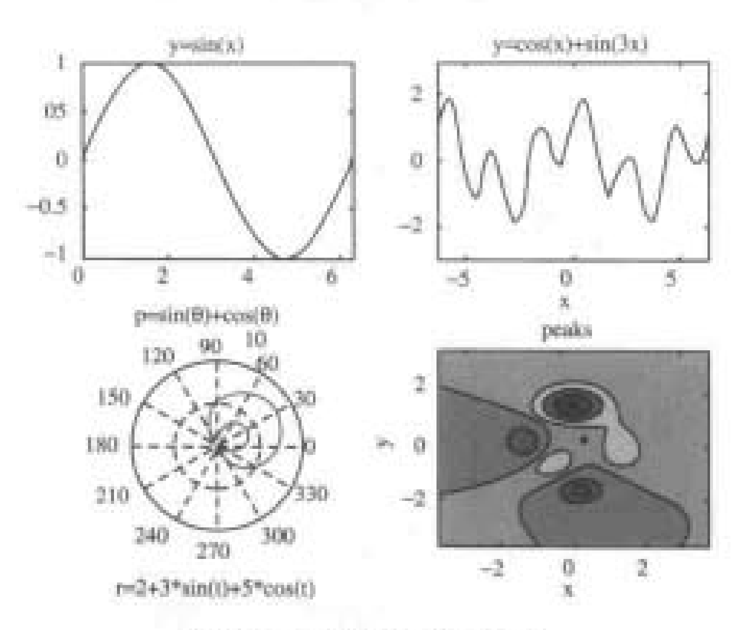



图 23-46　函数绘图（例 23-36）

## 23.5　图形窗口进阶

### 23.5.1　概述

MATLAB 图形窗口除了用于显示绘图函数的结果，还可以进行交互式绘图。MATLAB 交互式绘图工具包括三个面板：图形面板、绘图浏览器和属性编辑器，这些面板在默认视图下并不显示，打开这些面板的方法如表 23-10 所示。

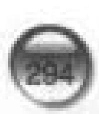
表 23-10　绘图工具面板显示方法

| 面　板 | 显示方法 | 其　他 |
|---|---|---|
| 图形面板（Figure Palette） | 命令 figurepalette；<br>或视图菜单下的 Figure Palette 项 | 图形工具条中的隐藏绘图工具按钮可以同时关闭这三个面板，显示绘图工具按钮可以同时显示这三个面板 |
| 绘图浏览器（Plot Browser） | 命令 plotbrowser；<br>或视图菜单下的 Plot Browser 项 | |
| 属性编辑器（Property Editor） | 命令 propertyeditor；<br>或视图菜单下的 Property Editor 项 | |

通过显示绘图工具按钮，打开三个面板之后的图形窗口如图 23-47 所示。

图形面板位于窗口左侧，通过图形面板的工具，用户可以创建和安排图形窗口下的子图分布，交互式地对工作区变量进行任意类型的图形绘制，或添加箭头、图框等标注元素。

图形浏览器位于窗口右侧，用于控制坐标轴或图形对象的显示、也可以通过 Add Data… 按钮在指定的坐标轴下添加数据进行新的附加绘图。



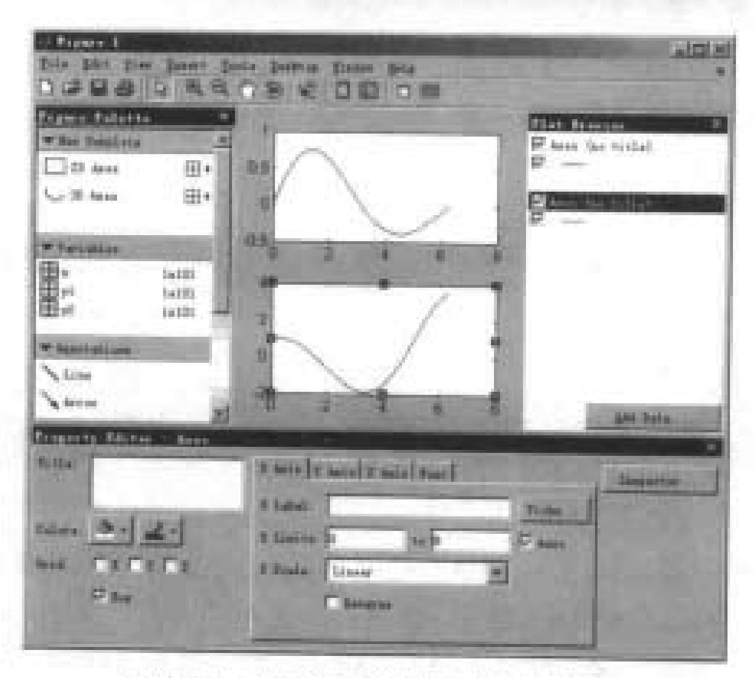


图 23-47 显示交互式绘图工具的图形窗口

属性编辑器位于窗口底部，当用户选择了某个图形对象时，属性编辑器中出现相应的各种常用属性的设置，如子图标题、网格、坐标轴标签、范围等。另外，单击 Inspector…按钮可以设置某个元素的所有属性。

本章后续部分将以一个完整的绘图实例说明这些面板的各种功能。该实例以例 23-37 所创建的数据为基础。

**例 23-37** 图形窗口进阶。

**解：** 在命令窗口输入：

```
>> x=0:0.05*pi:2*pi;
>> y1=exp(-0.2*x).*sin(5*x);
>> y2=exp(0.2*x).*cos(5*x);
>> plot(x,y1,x,y2)
```

## 23.5.2 图形面板

单击图形窗口 New Subplots 选项卡下的 2D Axes 按钮，会在当前绘图区的下方添加一行新的坐标轴；而单击右侧的田字方框和黑色箭头位置，则用户可以通过移动鼠标创建自定义行列的子图，当前已经存在的图形会被默认设置为编号最小的子图，并在窗口中以蓝色显示，待创建的子图以灰色显示，当用户选定好子图行列后，再次点击鼠标即可生效，其操作如图 23-48 所示，这会产生如图 23-49 所示的绘图区结果，其中已经存在的函数曲线是例 23-37 的代码产生的。

创建子图之后就可以在每一个子图区绘制函数了，这可以通过在图形面板的第二个选项卡中交互式地选择 MATLAB 工作空间中的变量，然后按用户指定的图形样式和绘图顺序来绘制函数曲线。

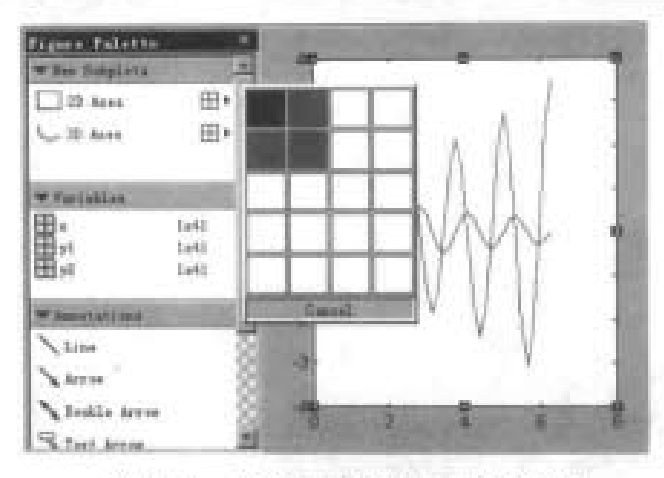



图 23-48　通过交互式绘图工具创建子图

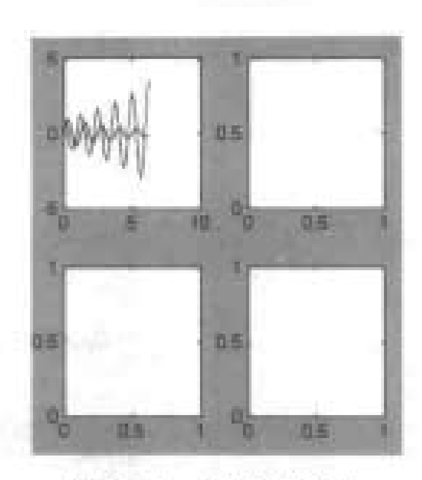


图 23-49　子图创建结果

一般先要选择坐标轴，然后按下 ctrl 键，用鼠标左键选择若干个参与绘图的变量，再点鼠标右键，从右键菜单中选择某种符合要求的绘图方式。

如图 23-50 中，选择了第二行第二例的子图（选中状态），然后利用 ctrl 选择了两个两个工作区变量，右键菜单中提供了一些简单的绘图项，如 plot(x,y1)等，更多自定义的绘图可以点击 More Plots…，出现如图 23-51 的绘图类型和参数选择窗口，用户也可以自定义绘图类型和绘图的参数。

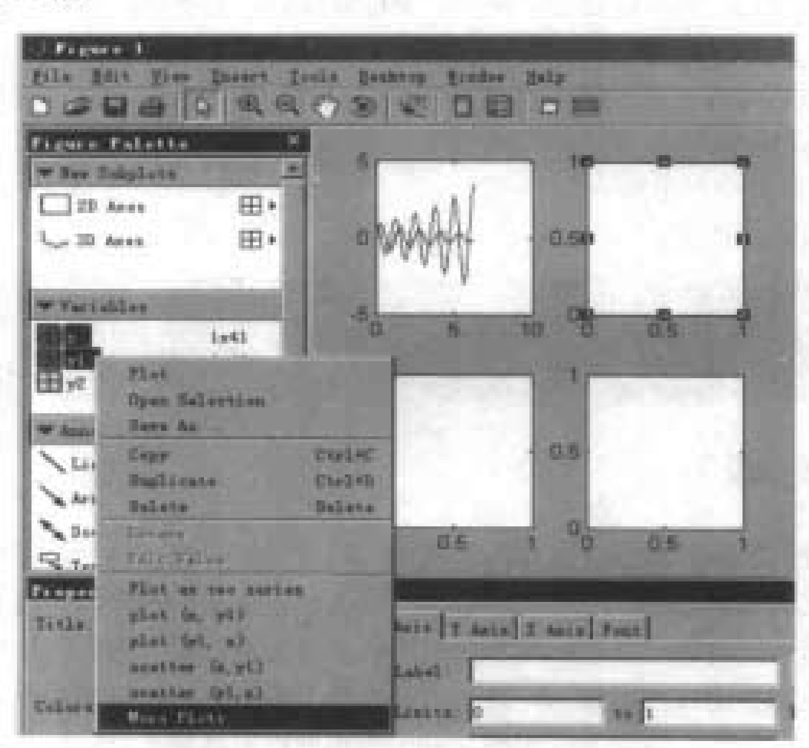



图 23-50　选择工作区变量进行交互式绘图

绘图类型可以设置为本章前面讲述的任何一种类型，如一般的线条图，或者各种特殊类型，然后在窗口最上方的文本框中可以设定绘图参数，实际上相当于绘图函数的输入参数。假如按照图 23-51 所示进行设置，则绘图结果如图 23-52 所示。



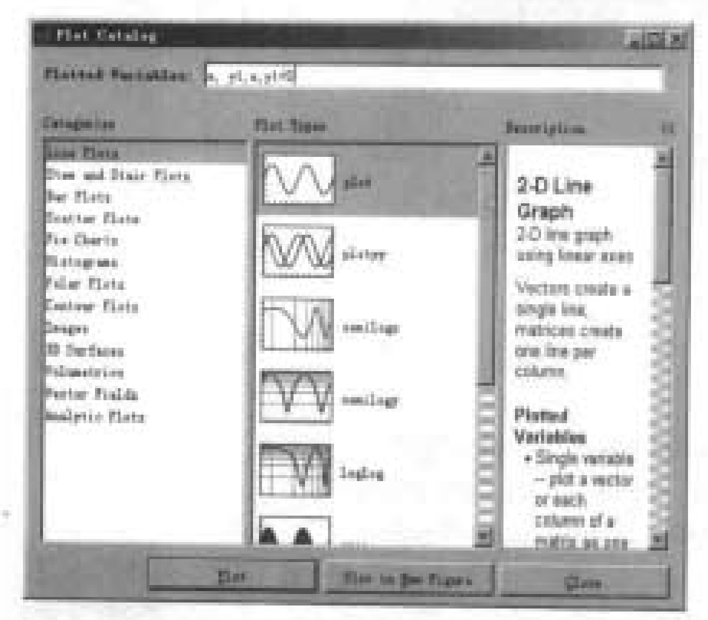



图 23-51 绘图类型和绘图参数设定窗口

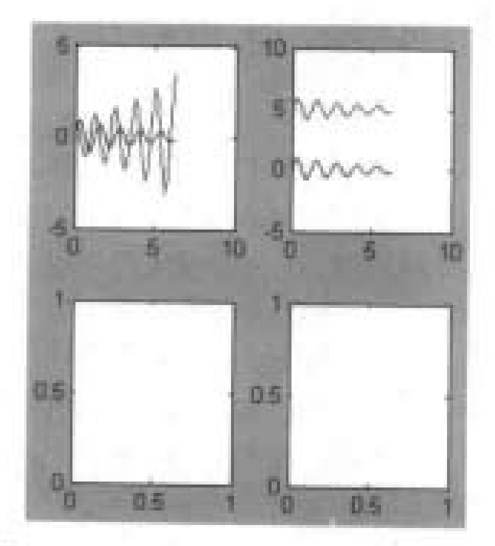


图 23-52 根据图 23-51 设置后的绘图结果

图形面板的最下面一个选项卡下的内容是用来进行图形标注的，包括线条箭头标注和图框标注。标注时，只需要选择相应的标注元素，在某个子图下用鼠标拖拽产生标注对象即可，操作简单，此处略过。

通过重复以上所述的绘图、标注操作，产生如图 23-53 所示的图形。

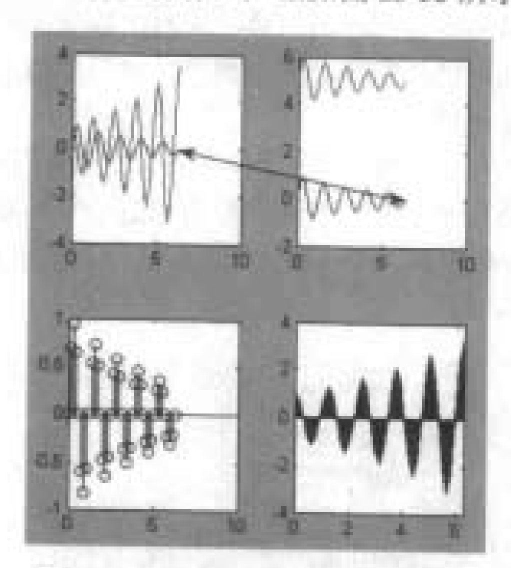


图 23-53 利用图形面板产生的绘图区

### 23.5.3 绘图浏览器

绘图浏览器用来显示当前绘图区中的所有坐标轴、图线，但不包括图形标注元素，用户可以通过绘图浏览器控制这些对象的显示和隐藏，可以在指定的坐标轴下添加绘图数据。

对应于图 23-53 的绘图区，绘图浏览器显示内容如图 23-54 所示。

图 23-54 中显示当前绘图区中所有元素：四个坐标轴，前两个坐标轴中分别绘制了两条曲线，第三个坐标轴中绘制了一个函数的火柴棒图，第四个坐标轴中绘制了面积图。

通过点击图 23-54 中的复选框，使其处于选中状态，则该图形元素（坐标轴或图线）会显示在绘图区，如使复选框处于非选中状态，则该图形元素将被隐藏。当某一个图形元素被选中时，对应的绘图区中该元素也处于选中待编辑状态，用户可以通过拖拽鼠标改变其尺寸、位置等，也可以通过下一小节要介绍的属性编辑器修改图形元素的各种属性。

当某一坐标轴元素被选中时，添加数据按钮（Add Data…）会处于可点击状态，这时候单击该按钮，就可以在选中的坐标轴上叠加绘图。在弹出的子窗口中，用户可以选择绘图数据和绘图类型。

图 23-54 绘图浏览器

### 23.5.4 属性编辑器

属性编辑器给用户修改图形元素（包括标注对象）的任意属性提供了一个便捷的图形界面的操作环境。当绘图区中某一元素（包括坐标轴、图线、各种标注对象、图例、颜色条等）被选中时，属性编辑器自动转换到选中元素的属性编辑界面。

以坐标轴对应的属性编辑器为例，用户可以编辑坐标轴标题，背景颜色、边框颜色，网格显示、边框显示，各坐标轴的标签、显示刻度、显示范围、线性坐标还是对数坐标、方向，以及文字属性设置等属性。

经过对坐标轴、图线的多次选择、编辑（注：可以配合绘图浏览器选择图形对象），可以将图 23-53 修饰成如图 23-55 所示，其中包括对标题、坐标轴边框颜色、网格线、文字属性、坐标轴显示范围、对数坐标、图线线型、面积图填充色和边界线颜色等属性的设置，请读者仔细地实践体会。

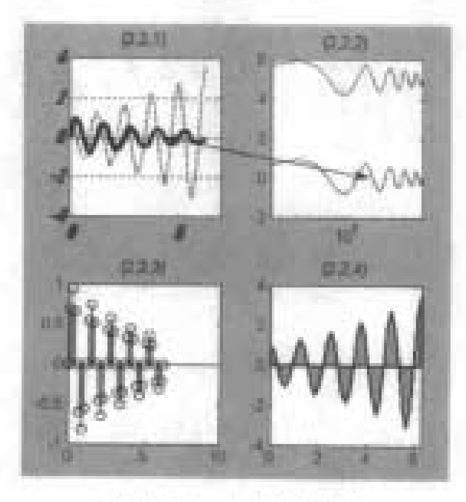


图 23-55 最终效果图

图 23-56 显示的是选中图 23-55 中第一个坐标轴时属性编辑器窗口内容，供读者参考。

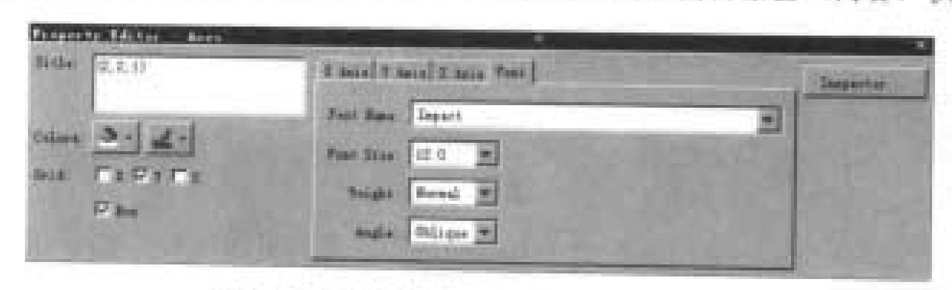


图 23-56 最终效果图对应的属性编辑器窗口

当然，通过单击 Inspector…按钮可以打开属性监视器界面，用户将可以编辑图形元素的任意属性，如图 23-57 所示。一般情况下，属性编辑器界面下提供的编辑项可以满足需要了。

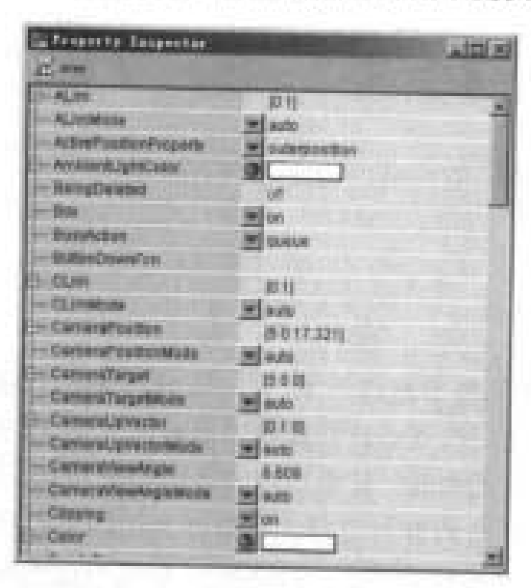


图 23-57 属性监视器界面

### 23.5.5 数据查视工具

当图形绘制完毕后，用户经常需要在查看图形局部细节和整体之间切换，这就需要便捷的数据查视工具，MATLAB 提供了常用的缩放、平移、旋转、摄像头工具等，方便用户进行各种数据查视的切换。

对于二维图形，只有缩放和平移工具。这些在默认视图下的图形工具条中都有对应的工具按钮。用户只需要选择相应的按钮，就可以在图形区通过鼠标拖拽产生缩放、平移效果。操作非常简单，不再赘述。不过需要注意的是，有时候几个子图绘制了相同的数据集合，并且通过箭头等标注元素将不同子图之间的特定点连接起来以达到数据指示的效果时，经常需要对标注元素进行锚定操作，否则，在用户使用这些数据查视工具变换图形显示效果时，标注元素不会随着坐标轴的缩放和平移进行相应的移动。



### 23.5.6 工作保存

下面简要介绍一下绘图工作的保存。

工作保存非常重要，MATLAB 中绘图结果的保存，最简单的方法是通过文件菜单（File）的几个保存选项。

（1）保存（Save）子菜单，可以将当前绘图区的绘图结果保存为二进制的.fig 文件，它只能由 MATLAB 来打开。

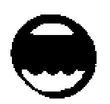
（2）另存为（Save as…）子菜单，可以设置保存文件格式，如可以设置为常用的.jpg，.bmp，.png，.tif 等格式，保存为这些格式的图形，可以用各种常用的图像处理软件，如 photoshop 等，进行修饰。

（3）产生 M-代码（Generate M-File…）子菜单，可以将当前绘图保存为 MATLAB 函数 M 文件，从而可以重复绘图，需要注意的是，产生的 M-代码中不保存当前绘图采用的数据集。

更多的关于图形保存方面的内容，请参考本书第 27 章。

## 23.6 小结

本章从介绍 MATLAB 的图形窗口入手，详细讲解了 MATLAB 中二维图形绘制的函数、工具、二维图形标注的方法，以及多种特殊绘图函数。其中 23.2 节的基本绘图函数和 23.3 节的图形标注是本章最重点的内容，其中讲到的各种函数的常见用法，读者应该熟悉掌握；23.5 节中讲解的图形窗口进阶内容，可以大大降低绘图的门槛和提高绘图的效率，读者也应该熟练应用；至于 23.4 节的特殊绘图，在不同的专业绘图领域有不同的要求和应用，读者可以选择性学习，但本章所有例子都只用到了简单的绘图函数和标注函数的组合，都是二维绘图中最基本最经典的实例，读者应该仔细阅读体会，最好逐一实践练习。



# 第 24 章 三维图形

MATLAB 中可以通过二维或三维图形实现数据的可视化。本章紧接前文，介绍 MATLAB 中在三维空间上实现数据可视化的方法，这包括一般的三维曲线、曲面图形和三维片块模型。此外，本章还将介绍三维视图相关的工具，包括视角设置和数据查视工具。

## 24.1 创建三维图形

### 24.1.1 三维图形概述

MATLAB 中的三维图形包括三维曲线图、三维网格线图和三维表面图。创建三维图形和创建二维图形的过程类似，都包括数据准备、绘图区选择、绘图、设置和标注，以及图形的打印或输出。不过，三维图形能够设置和标注更多的元素，如颜色过渡、光照和视角等。

MATLAB 中创建三维图形的基本流程如表 24-1 所示。

表 24-1 三维绘图基本流程

| 三维绘图基本流程 | M-代码举例 | 备 注 |
|---|---|---|
| 1. 数据准备 | x=-8:0.1:8;<br>y=-8:0.1:8;<br>[X,Y]=meshgrid(x,y);<br>Z=(exp(X)-exp(Y)).*sin(X-Y); | 三维曲线图用一般的数组创建即可<br>三维网线图和三维表面图的创建需要通过 meshgrid 创建网格数据 |
| 2. 图形窗口和绘图区选择 | figure | 创建绘图窗口和选定绘图子区 |
| 3. 绘图 | surf(X,Y,Z) | 创建三维曲线图、网线图或表面图 |
| 4. 设置视角 | view([75 25]) | 设置观察者查看图形的视角和 Camera 属性 |

续表

| 三维绘图基本流程 | M-代码举例 | 备　注 |
|---|---|---|
| 5．设置颜色表 | colormap hsv<br>shading interp | 为图形设置颜色表，从而可以用颜色显示 z 值的大小变化<br>对表面图和三维片块模型还可以设置颜色过渡模式 |
| 6．设置光照效果 | light('Position',[1 0.5 0.5])<br>lighting gouraud<br>material metal | 设置光源位置和类型<br>对表面图和三维片块模型还可以设置反射特性 |
| 7．设置坐标轴刻度和比例 | axis square<br>set(gca,'ZTickLabel',") | 设置坐标轴范围、刻度和比例 |
| 8．标注图形 | Xlabel('x')<br>Ylabel('y')<br>colorbar | 设置坐标轴标签、标题等标注元素 |
| 9．保存、打印或导出 | print | 将绘图结果打印或导出为标准格式图像 |

从表 24-1 可以看出，三维绘图中多了颜色表、颜色过渡、光照等专门针对三维图形的设置项，其他基本流程都和二维绘图类似。

表 24-1 中举例的 M-代码连贯起来运行，可以得到如图 24-1 所示的绘图结果。

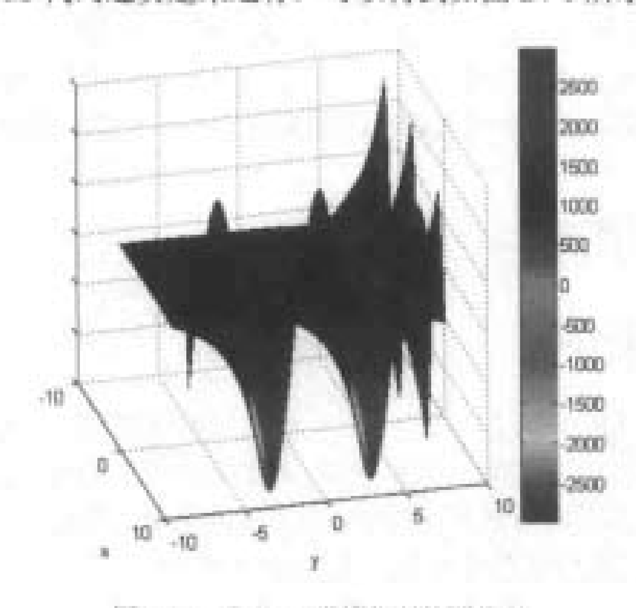



图 24-1　表 24-1 举例代码绘图结果

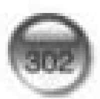
下面几小节介绍各种类型的三维图形绘制的方法。

## 24.1.2　三维曲线图

三维曲线图描述的是 *x*，*y* 沿着一条平面曲线变化时，*z* 随之变化的情况。MATLAB 中三维曲线图的绘制函数是 plot3，其常见调用格式是：

plot3(*x*,*y*,*z*)

其中 *x*、*y*、*z* 是三个尺寸相同的数组。plot3 和 plot 类似，也可以用表征线型和颜色的





字符串来设置三维曲线的线型和颜色，或者通过设置 LineStyle、LineWidth、Marker 属性来定义曲线线型、线宽和数据点标记等。

一般情况下，*x*、*y*、*z* 是具有同样长度的一维数组，这时候 plot3 画出一条三维曲线。实际上，*x*、*y*、*z* 也可以是同样尺寸且具有多列的二维数组，这时候 plot3 会将 *x*、*y*、*z* 对应的每一列当作一组数据分别绘制出多条曲线。

例 24-1 plot3 绘制三维曲线图。

**解**：在命令窗口中输入：

```
%Ex24-1 plot3
close all
x=-5:0.4:5;
y=5:-0.4:-5;
z=exp(-0.2*x).*cos(y);
[X,Y]=meshgrid(x,y);
Z=exp(-0.2*X).*cos(Y);
figure
subplot(2,1,1)
plot3(x,y,z,'or',x,y,z)
subplot(2,1,2)
plot3(X,Y,Z)
```

此 M 文件执行后，绘图结果如图 24-2 所示。

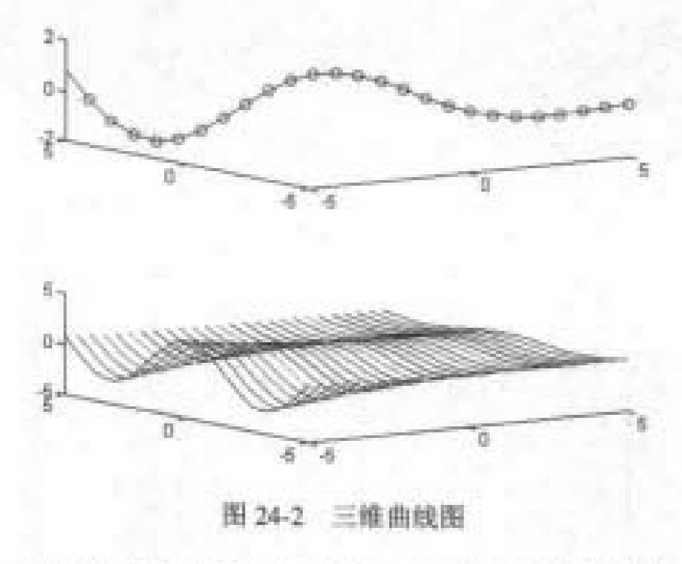


图 24-2 三维曲线图

从图 24-2 的第二个子图中可以看到，MATLAB 在同时绘制多条三维曲线时，会通过默认的颜色序，区分各条曲线，这些都和二维图形的绘制是一样的。

### 24.1.3 三维曲面图

#### 1．矩形网格

（*x*, *y*）定义在一条曲线上时，可以用曲线图描绘（*x*, *y*, *z*）的变化关系。对于（*x*, *y*）定义在一个区域中的情况，则应该用曲面图来显示。MATLAB 中的曲面图分为网线图和表

面图两种类型。

MATLAB 中通过矩形网格组合来描绘曲面，即将（*x*, *y*）定义的区域分解为一个个小矩形区域，然后计算在这个小矩形区域每一个顶点处的 *z* 值，在显示时通过把这些邻近的顶点都互相连接起来，从而组合出整个（*x*, *y*）区域上的（*x*, *y*, *z*）曲面。

在组合这些网格显示整个曲面时，MATLAB 可以采用两种方式：

（1）一种是只用线条将各个邻近顶点连接，而网格区域内部显示为空白，这种通过矩形网格边框线来显示整个曲面的曲面图称为网线图；

（2）另一种则不但显示网格线边框，而且将其内部填充着色，从而通过一个个矩形平面来组合显示整个曲面，这种曲面图称为表面图。

因此，绘制三维曲面图，首先要创建（*x*, *y*）的网格，MATLAB 提供了 meshgrid 函数可以在（*x*, *y*）的矩形区域上创建网格，meshgrid 的调用格式为：

$$[X, Y]=\text{meshgrid}(x, y)$$

通过数据重复在一维数组 ***x***、***y*** 的每一个交叉点上创建网格点，当 ***x*** 和 ***y*** 都是长度为 ***n*** 的一维数组时，则 ***X*** 和 ***Y*** 是 ***n*n*** 的二维数组，每一个对应的（***X***, ***Y***）就是一个网格点。

**例 24-2** 矩形网格。

**解**：在命令窗口中输入：

```
%Ex24-2 meshgrid
x=-5:0.5:5;
y=5:-0.5:-5;
[X,Y]=meshgrid(x,y);
plot(X,Y,'o')
```

运行这段 M 代码则绘制出如图 24-3 所示的矩形网格顶点。

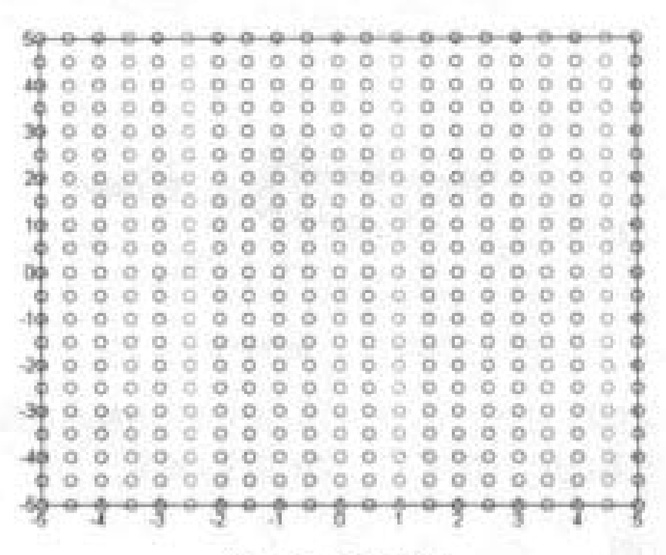



图 24-3 矩形网格

运行 whos 查看工作区变量属性，得到结果为：

```
  Name      Size                    Bytes  Class
```



```
  X        21x21                  3528  double array
  Y        21x21                  3528  double array
  x         1x21                   168  double array
  y         1x21                   168  double array

Grand total is 924 elements using 7392 bytes
```

创建了（*X*,*Y*）网格点后就可以绘制网线图或表面图了。

2. 三维网线图

网线图是把邻近的网格顶点（*X*, *Y*）对应的曲面上的点（*X*, *Y*, *Z*）用线条连接起来的三维曲面图，网格对应的曲面区域内则显示为空白。

MATLAB 中可以通过 mesh 函数绘制三维网线图，该函数的语法格式为：

mesh(*X*,*Y*,*Z*)

绘制网格点数据（*X*,*Y*,*Z*）对应的三维曲面的网线图，其中 ***X***、***Y*** 一般是通过[*X*,*Y*]=meshgrid(*x*,*y*)生成的 ***n*n*** 二维数组，***Z*** 是通过函数对应关系由 ***X***、***Y*** 计算生成的函数值。

另外，MATLAB 中还有两个 mesh 的派生函数：

（1）meshc 在绘制网线图的同时，在 *x*-*y* 平面上绘制函数的等值线；

（2）meshz 则在网线图基础上在图形的底部外侧绘制平行 *z* 轴的边框线。

**例 24-3** 三维网线图。

**解**：在命令窗口中输入：

```
%Ex24-3 mesh
close all
clear
[X,Y] = meshgrid(-3:.5:3);
Z = 2*X.^2-3*Y.^2;
subplot(2,2,1)
plot3(X,Y,Z)
title('plot3')
subplot(2,2,2)
mesh(X,Y,Z)
title('mesh')
subplot(2,2,3)
meshc(X,Y,Z)
title('meshc')
subplot(2,2,4)
meshz(X,Y,Z)
title('meshz')
```

运行这段 M 代码，得到图 24-4 的绘图结果。

从图 24-4 可以看到，plot3 只能画出 ***X***、***Y***、***Z*** 的对应列表示的一系列三维曲线，它只要求 ***X***、***Y***、***Z*** 三个数组具有相同的尺寸，并不要求（***X***, ***Y***）必须定义网格点。

mesh 函数则要求（***X***, ***Y***）必须定义网格点，并且在绘图结果中可以把邻近网格点对应的三维曲面点（***X***, ***Y***, ***Z***）用线条连接起来。

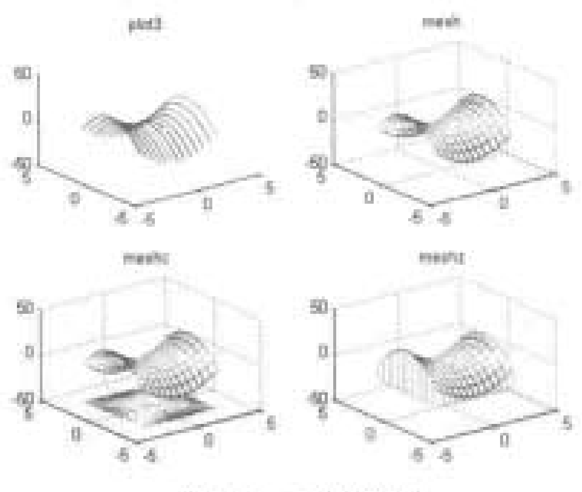



图 24-4　三维网线图

此外，plot3 绘图时按照 MATLAB 绘制图线的默认颜色序，循环使用颜色区别各条三维曲线，而 mesh 绘制的网线图中颜色用来表征 z 值的大小，可以通过 colormap 命令显示表示图形中颜色和数值对应关系的颜色表。

### 3. 三维表面图

三维表面图和三维网线图不同，其显示结果中用黑色的线段连接邻近曲面点（*X*, *Y*, *Z*），而对网格区域内的曲面区域则用颜色填充。

MATLAB 中绘制三维表面图的函数是 surf，其用法和 mesh 一样。另外，surfc 可以在表面图的 *x*-*y* 平面上附加绘制等值线；surfl 可以给表面图添加光照效果。

**例 24-4**　三维表面图。

**解**：在命令窗口中输入：

```
%Ex24-4 surf
close all
clear
[X,Y] = meshgrid(-3:.5:3);
Z = 2*X.^2-3*Y.^2;
subplot(2,2,1)
mesh(X,Y,Z)
title('mesh')
subplot(2,2,2)
surf(X,Y,Z)
title('surf')
subplot(2,2,3)
surfc(X,Y,Z)
title('surfc')
subplot(2,2,4)
surfl(X,Y,Z)
title('surfl')
```

以上代码运行结果如图 24-5 所示。





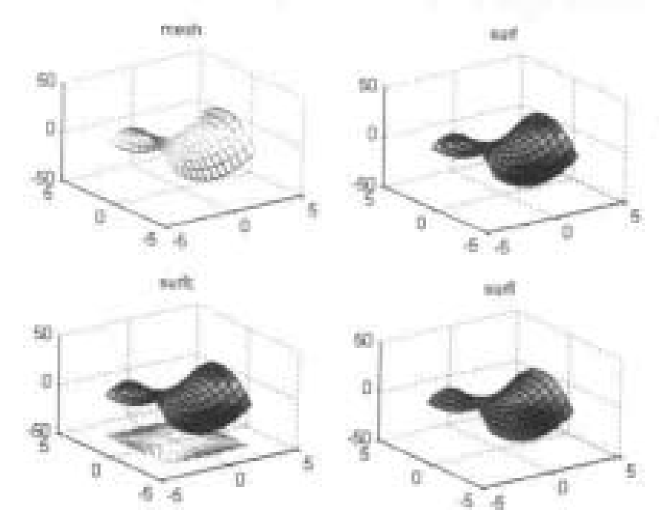



图 24-5 三维表面图

从图 24-5 中，可以看到 mesh 和 surf 绘图的区别。mesh 绘制的网线图中网格边框线是彩色的，区域内是空白的，线条颜色表示了该处 $z$ 值的大小；而 surf 绘制的表面图中，网格边框线是黑色的，而区域内是彩色填充的，区域颜色表示该处 $z$ 值的大小。这两种曲面图中都可以通过 colormap 显示颜色和 $z$ 值的对应表。

另外，网线图和表面图都支持颜色过渡和光照等效果，这部分内容见本书第 25 章。

#### 4. 网格边框线设置

默认情况下，mesh 绘制的三维网线图中，观察者所处位置不可见的网格边框线会自动被隐藏。用户可以通过设置 hidden 开关，设置是否显示这些不可见的网格边框线，hidden off 会设置禁止自动隐藏，从而在 mesh 图中用显示这些边框线。

**例 24-5** 网格边框线设置。

**解**：在命令窗口中输入：

```
%Ex24-5 hidden close all
clear
[X,Y] = meshgrid(-3:.25:3);
Z = -sqrt(X.^2+3*Y.^2);
subplot(1,2,1)
mesh(X,Y,Z)
hidden on
title('hidden on')
subplot(1,2,2)
mesh(X,Y,Z)
hidden off
title('hidden off')
```

以上代码运行结果如图 24-6 所示。

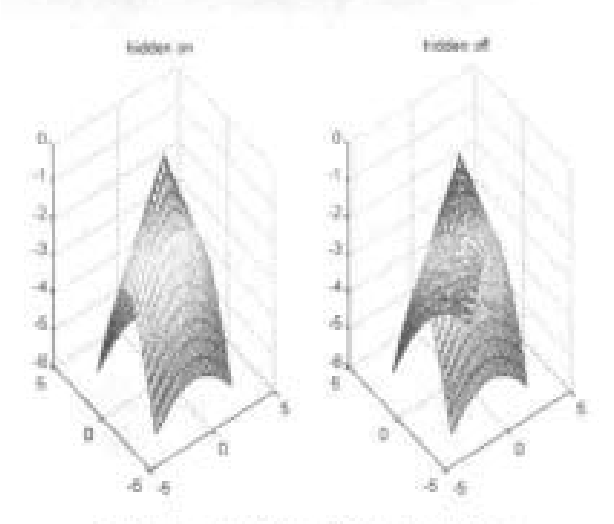



图 24-6　网格边框线的隐藏和显示

### 5．非网格数据点绘图

mesh 和 surf 绘制三维曲面图，都要求数据（*X*,*Y*）是均匀分布的网格点坐标，但很多实际采样得到的数据是散乱分布的，这时候就需要通过 meshgrid 先创建插值网格点，并在这些网格点上插值计算 *z* 值，这样就可以用 mesh 或 surf 命令绘制三维曲面图了。

MATLAB 中，在网格点上插值计算 *z* 值的函数是 griddata，其常用调用格式是：

**ZI** = griddata(***x***,***y***,***z***,***XI***,***YI***,method)

其中 ***x***、***y***、***z*** 是采样得到的原始数据点，即插值源数据，***XI***、***YI*** 是待插值数据点坐标，method 指定了插值方法，返回值 ***ZI*** 是在（***XI***,***YI***）处的函数插值结果。

method 字段的可选字符串有：'linear'（线性插值算法，默认的插值算法）、'cubic'（立方插值算法）、'nearest'（最邻近点插值算法）和'v4'（MATLAB 4 网格点插值法）。

其中，'cubic'和'v4'插值算法得到的插值曲面连续光滑，而'linear'和'nearest'则不连续，默认情况下 MATLAB 会采用'linear'线性插值算法。

**例 24-6**　非网格数据点绘图。

**解**：在命令窗口中输入：

```
%Ex24-6 non-uniform data
x=rand(1,20);
y=rand(1,20);
z=cos(0.5.*x).*sin(y);
xi=linspace(0,1,50);
yi=linspace(0,1,50);
[X,Y]=meshgrid(xi,yi);
subplot(2,2,1)
Z1=griddata(x,y,z,X,Y,'linear');
mesh(X,Y,Z1)
title('linear')
subplot(2,2,2)
Z1=griddata(x,y,z,X,Y,'nearest');
mesh(X,Y,Z1)
title('nearest')
subplot(2,2,3)
```



```
Z1=griddata(x,y,z,X,Y,'cubic');
mesh(X,Y,Z1)
title('cubic')
subplot(2,2,4)
Z1=griddata(x,y,z,X,Y,'v4');
mesh(X,Y,Z1)
title('v4')
```

代码运行结果如图 24-7 所示。

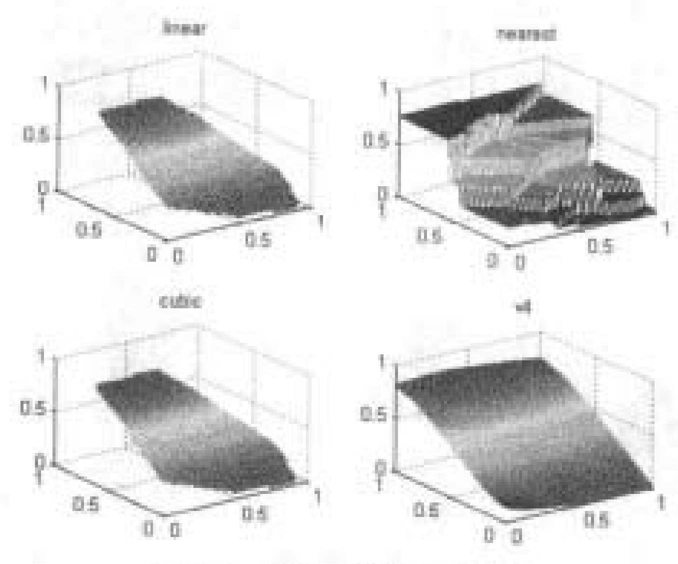



图 24-7　非网格数据点插值绘图

### 24.1.4　特殊三维绘图

MATLAB 中也可以创建柱状图、散点图、饼状图等特殊样式的三维图形。

**1. 柱状图**

三维柱状图绘制函数是 bar3 和 bar3h，用法和 bar，barh 类似，将每一元素用一个三维条柱图示。

**例 24-7**　三维柱状图。

**解：**在命令窗口中输入：

```
%Ex24-7 bar3 bar3h
clear
x=rand(3,10);
subplot(2,2,1)
bar(x)
title('bar')
subplot(2,2,2)
barh(x,'stack')
title('barh-stack')
subplot(2,2,3)
bar3(x)
title('bar3')
subplot(2,2,4)
```

```
bar3h(x,'stack')
title('bar3h-stack')
```

结果如图 24-8 所示。

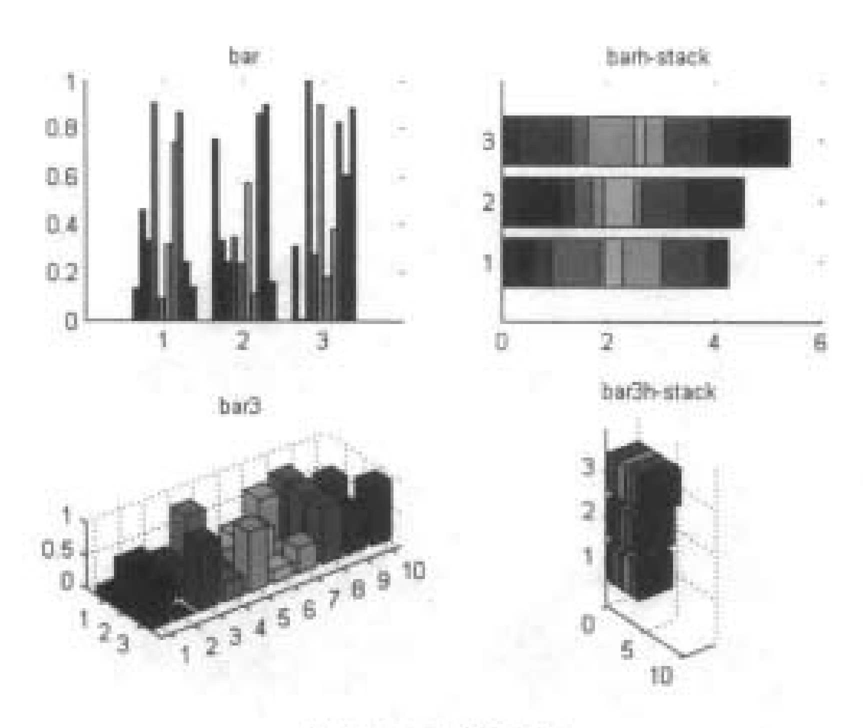



图 24-8　三维柱状图

从图 24-8 可以看出，对于普通的二维数组数据，二维柱状图把每一行的数据元素表示在一组内，用同一颜色标记不同行之间相同列的数据，而三维柱状图则把每一列的数据元素表示在一组内，因此在 bar3 绘制的三维柱状图中不但可以清晰地比较各行内元素的差别，也能清晰地看到各列内元素值的差别。

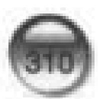
**2. 散点图**

三维散点图绘制函数是 scatter3。和 scatter 类似，scatter3 将三维空间的离散点（x,y,z）标示在三维坐标轴下，实际上和指定标记点类型的 plot3 结果一样。

**例 24-8**　三维散点图。

**解**：在命令窗口中输入：

```
%EX24-8 scatter3
close all
clear
x=rand(1,10);
y=rand(1,10);
z=x.^2+y.^2;
scatter3(x,y,z,'ro')
hold on
[X,Y]=meshgrid(0:0.1:1);
Z=X.^2+Y.^2;
mesh(X,Y,Z)
hidden off
```





代码运行后绘图结果如图 24-9 所示。

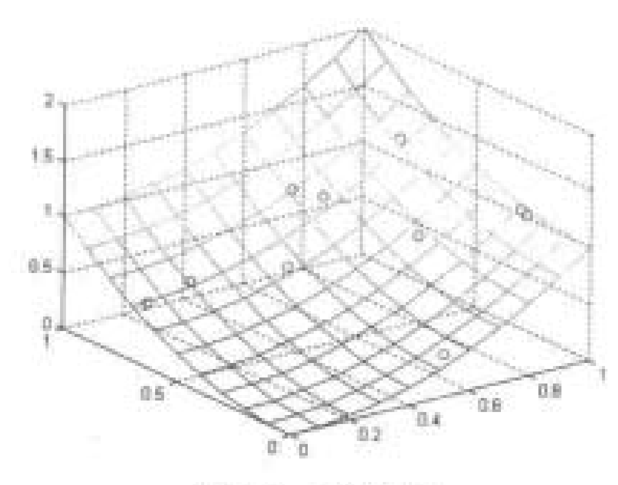


图 24-9　三维散点图

在例 24-8 中，scatter3(*x*,*y*,*z*,'ro')指定了标记散点为红色圆圈，hold on，mesh(*X*,*Y*,*Z*)、hidden off 则在此散点图基础上附加绘制了网线图，并设置网格线不隐藏模式（这样在网线图表示的曲面后的散点就会显示在图形中）。

### 3. 饼状图

三维饼状图的绘制函数是 pie3，用法和 pie 类似，以三维饼状图形显示各组分所占比例。

**例 24-9**　三维饼状图。

**解**：在命令窗口中输入：

```
%Ex24-9 pie3
x=[32 45 11 76 56];
explode=[0 0 1 0 1];
pie3(x,explode)
```

代码运行后绘制结果如图 24-10 所示。

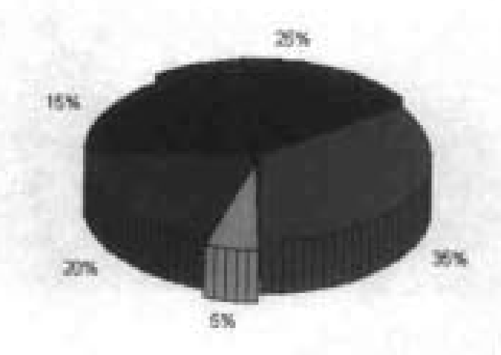



图 24-10　三维饼状图

### 4. 火柴杆图

三维火柴杆图的绘制函数是 stem3，用法和 stem 类似，stem3(*x*,*y*,*z*)在三维坐标轴下（*x*,*y*）

处绘制长度为 z 平行于 z 轴的的火柴杆。

**例 24-10** 三维火柴杆图。

**解：** 在命令窗口中输入：

```
%Ex24-10 stem3
clear
x=rand(1,10);
y=rand(1,10);
z=x.^2+2*y;
stem3(x,y,z,'fill')
```

代码运行后绘图结果如图 24-11 所示。

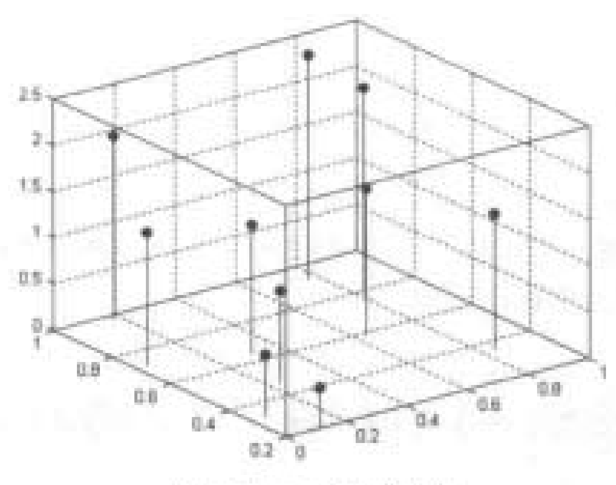


图 24-11　三维火柴杆图

### 5. 向量场图

三维向量图绘制函数是 quiver3，用法和 quiver 类似，quiver3 可以在三维空间的指定点用箭头标示指定的向量。

**例 24-11** 三维向量场图。

**解：** 在命令窗口中输入：

```
%Ex24-11 quiver3
clear
close all
[X,Y]=meshgrid(-3:0.4:3);
Z=-3*X.^2-Y.^2;
[U,V,W]=surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W,0.2)
hold on
surf(X,Y,Z)
```

代码运行后绘图结果如图 24-12 所示。

例 24-11 中，surfnorm 函数计算曲面在（$X,Y,Z$）处的法向量（$U,V,W$），然后通过三维向量场图绘制函数 quiver3($X,Y,Z,U,V,W$,0.2)把（$U,V,W$）缩减为原来的 0.2 倍长度后，绘制在（$X,Y,Z$）处，最后叠加绘制了（$X,Y,Z$）的三维表面图。



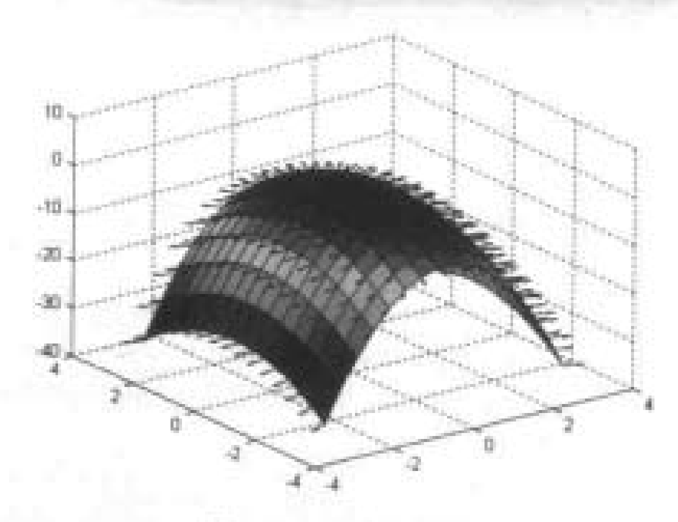


图 24-12　三维向量场图

### 6. 等值线图

MATLAB 中，三维等值线图绘图函数是 contour3，它不同于二维等值线图那样只在 x-y 平面上显示 z 值的等值圈，而是在把等值线显示在平行于 x-y 平面的每一个切面上。

**例 24-12**　三维等值线图。

**解**：在命令窗口中输入：

```
%Ex24-12 contour3
clear
close all
[X,Y]=meshgrid(-3:0.01:3);
Z=X.^2+Y.^2;
contour3(X,Y,Z,20)
view([45 50])
```

绘图结果如图 24-13 所示。

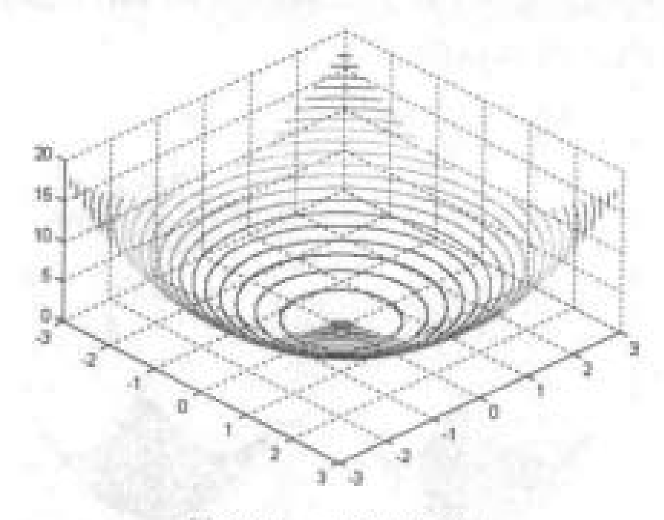


图 24-13　三维等值线图

例 24-12 的代码中 view([45 50])是用来设置视角的，本章后面会介绍到。

7．简易绘图函数

MATLAB 中，还有一些简易绘制三维图形的函数，它们接收函数句柄作为输入参数，能快捷地绘制二元函数图形，如表 24-2 所示。

表 24-2　三维图形简易绘制函数

| 函　数 | 说　明 |
|---|---|
| ezplot3(funx,funy,funz,[tmin,tmax]) | 在[tmin,tmax]范围下绘制(funx(t),funy(t),funz(t))三维曲线 |
| ezmesh (fun,domain) | 在 domain 指定的区域绘制 fun 指定的二元函数的网线图 |
| ezmeshc (fun,domain) | 在 domain 指定的区域绘制 fun 指定的二元函数的网线图，并在 x-y 平面叠加绘制等高线 |
| ezsurf (fun,domain) | 在 domain 指定的区域绘制 fun 指定的二元函数的表面图 |
| ezsurfc (fun,domain) | 在 domain 指定的区域绘制 fun 指定的二元函数的表面图，并在 x-y 平面叠加绘制等高线 |

表 24-2 中，funx，funy，funz，fun 这些参数可以是函数句柄、匿名函数或者函数字符串，domain 是指定平面区域[xmin,xmax,ymin,ymax]的数组。

**例 24-13**　简易三维绘图函数。

**解：**在命令窗口中输入：

```
%Ex24-13 easy 3-D plot
close all
clear
subplot(2,2,1)
ezplot3('sin(t)','cos(t)','sin(2*t)',[0,2*pi])
subplot(2,2,2)
ezmesh(@peaks,[-5 5 -5 5])
subplot(2,2,3)
ezsurf(@(x,y)(x.^2+y.^2),[-5 5 -5 5])
subplot(2,2,4)
ezsurfc(@(x,y)(x.^2+y.^2),[-5 5 -5 5])
```

代码运行后绘图结果如图 24-14 所示。

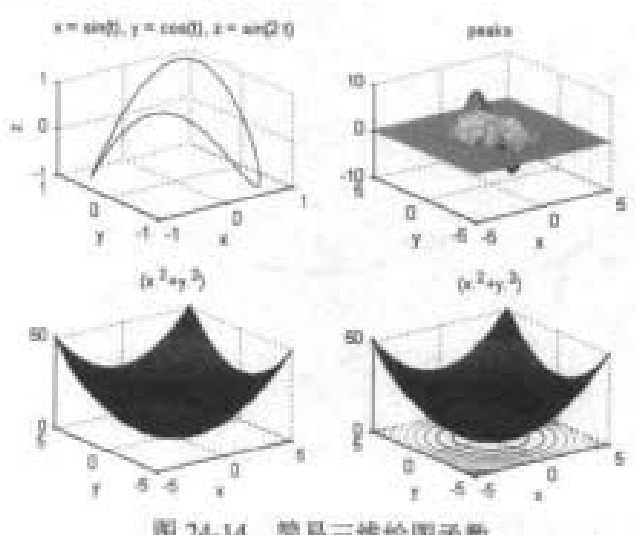
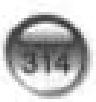



图 24-14　简易三维绘图函数





例 24-13 中，peaks 是 MATLAB 内置的一个二元函数，因此@peaks 代表这一函数的句柄，和@sin、@cos 类似。

## 24.2　创建三维片块模型

### 24.2.1　创建片块模型

MATLAB 中三维可视化技术还包括创建片块模型，和一般的三维曲线、曲面图相比，片块模型能更逼真地描绘真实世界的实体，而三维曲线、曲面图更适合描绘二元函数。

MATLAB 中的片块模型实际上是一系列互相连接的多边形，其创建函数如表 24-3 所示。

表 24-3　片块模型创建函数

| 函　数 | 说　明 |
|---|---|
| fill(*X*,*Y*,*C*) | 创建 *X*、*Y* 指定顶点坐标的平面多边形，用 *C* 指定的颜色填充为片块模型 |
| fill3(*X*,*Y*,*Z*,*C*) | 创建 *X*、*Y*、*Z* 指定顶点坐标的空间多边形，用 *C* 指定的颜色填充为片块模型 |
| patch(*X*,*Y*,*Z*,*C*) | patch 的高级语法形式：<br>创建 *X*、*Y*、*Z* 指定顶点坐标的空间多边形，用 *C* 指定的颜色填充为片块模型 |
| patch('PropertyName',PropertyValue) | patch 的低级语法形式：<br>通过对待创建的片块模型的必须属性 PropertyName 赋值为 PropertyValue 来创建片块模型 |

fill、fill3 和 patch 的高级语法形式都是通过设置顶点坐标定义片块多边形。而通过 patch 的低级语法形式则可以实现更精细的设置，表 24-4 列出了通过这种方法创建片块模型时必须设置的片块属性项。

表 24-4　创建片块模型必须属性项

| 属　性 | 取值范围 | 说　明 |
|---|---|---|
| XData | 一维或二维数组 | 指定片块多边形顶点的 *X* 坐标，每一列代表一个多边形片块模型 |
| YData | 一维或二维数组 | 指定片块多边形顶点的 *Y* 坐标，每一列代表一个多边形片块模型 |
| ZData | 一维或二维数组 | 指定片块多边形顶点的 *Z* 坐标，每一列代表一个多边形片块模型 |
| Vertices | *k**3 数组或 *k**2 数组 | 指定具有 *k* 个顶点的片块多边形的 *k* 个顶点的坐标，每一行代表一个顶点 |
| Faces | *m***n* 数组 | 指定 *m* 个片块多边形各自的顶点连接顺序，每一行表示一个片块多边形，顺次给出多边形顶点索引（即 Vertices 中的行标） |

**例 24-14**　创建片块模型。

**解**：在命令窗口中输入：

```
%Ex24-14 create single patch
clear
close all
subplot(2,2,1)
x=[0 1 0.5];y=[0 0 1];
fill(x,y,'r');title('fill')
```

```
subplot(2,2,2)
X=[0 0 0 0];Y=[0 1 1 0];Z=[0 0 1 1];
fill3(X,Y,Z,'g');title('fill3')
subplot(2,2,3)
patch(X,Y,Z,'b');view([30,30])
title('patch:high-level syntax')
subplot(2,2,4)
patch('XData',X,'YData',Y,'ZData',Z);view([30,30])
title('patch:low-level syntax')
```

代码运行结果如图 24-15 所示。

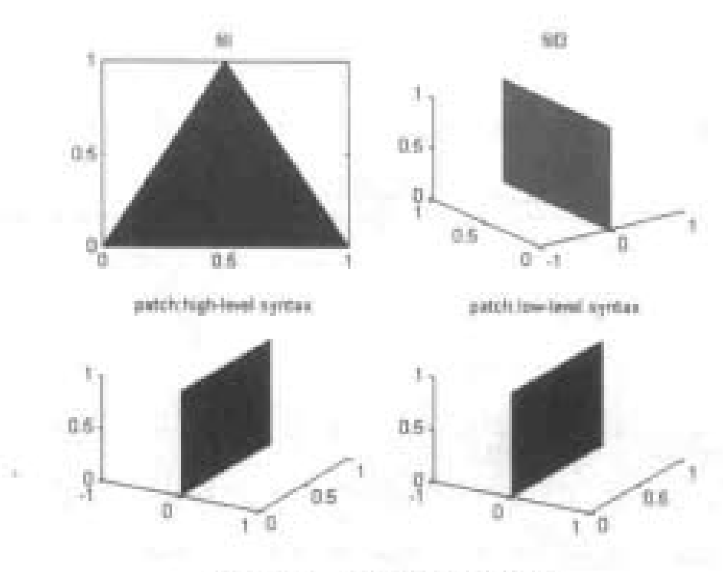



图 24-15　创建三维片块模型

在创建单个片块模型时，每个顶点坐标最好只出现一次，MATLAB 会自动识别是否需要把最后一个顶点和第一个顶点连接起来。

另外，创建片块时，一般要求用户自己指定顶点、边界线和区域填充色，这在 fill，fill3 和 patch 的高级语法调用中都是必须的，而在 patch 的低级语法形式中，虽然不必指定颜色，但当用户不指定时，MATLAB 默认用黑色边框线和白色填充色。

### 24.2.2　多个片块模型的创建和颜色设置

创建片块模型时，如果输入的坐标数组是普通的二维数组，那么 MATLAB 会自动把每一列的数组当作一个片块多边形的顶点处理，因此对于 $m*n$ 的坐标数组，将会创建 $n$ 个分别具有 $m$ 个顶点的片块模型。这样的方法要求创建每一个片块时都输入一次顶点坐标，有些情况下，创建的多个片块会互相共用一些顶点，这种情况下如果还要求对每一个面都指定各个顶点的坐标将会很繁琐，通过 patch 可以大大降低输入的工作量，patch 的低级语法形式为：

patch('Vertices',vmatrix,'Faces',fmatrix)

其中 vmatrix 数组指定所有出现的顶点的坐标，fmatrix 数组指定各个多边形片块的顶





点索引（即在 vmatrix 数组中的行标）。fmatrix 的每一行确定一个多边形片块。

**例 24-15** 创建多个片块模型。

**解**：在命令窗口中输入：

```
%Ex24-15 draw multi-patches
clear
close all
subplot(1,2,1)
X=[0 0;0 1;0 1;0 0];
Y=[0 0;1 0;1 1;0 1];
Z=[0 0;0 0;1 0;1 0];
patch(X,Y,Z,'r')
view([30,30])
subplot(1,2,2)
Vm=[0 0 0;1 0 0;1 0 1;0 0 1;0 1 1;1 1 1;1 1 0];
Fm=[1 2 3 4;3 4 5 6;6 3 2 7];
patch('Vertices',Vm,'Faces',Fm,'EdgeColor','r')
view([30,30])
```

代码运行后绘图结果如图 24-16 所示。

从图 24-16 中可以看到，创建的多个片块模型填充色是一样的，这显然不能满足形象显示真实世界实体的需要。实际上，MATLAB 中可以通过 patch 低级语法格式设置片块模型显示属性，达到复杂模拟的效果。表 24-5 列出了和片块显示效果相关的一些属性项。

大多数多片块模型共存的情况下，用户都需要设置边界或区域内以插值过渡色显示，这时可以设置 CData 和 FaceVertexCData 属性中的一个，然后将 EdgeColor 和（或）FaceColor 属性设置为 interp。这种情况下，MATLAB 对 CData 和 FaceVertexCData 属性值的解释比较复杂，解释方式依赖于该属性设置值的类型和数组形状。

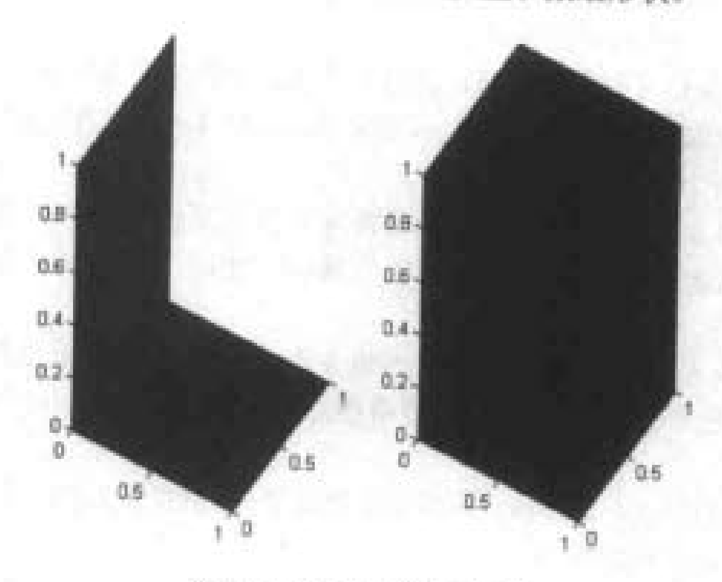



图 24-16 创建多个片块模型

表 24-5 片块显示效果属性

| 属　性 | 取值范围 | 说　明 |
| --- | --- | --- |
| Marker | 点标记符号，如'o' | 标记原始数据点（顶点） |
| LineStyle | 线型标记符号，如'-.' | 设置片块多边形边界线线型（设为 none 则不显示边界线） |
| LineWidth | 线宽数值 | 设置边界线线宽 |
| CData | 标量或数组 | 根据取值类型指定顶点、边界线、界面区域的颜色 |
| FaceVertexCData | 标量或数组 | 根据取值类型指定顶点、界面区域的颜色 |
| EdgeColor | 表示颜色的数组或字符，或 none，flat，interp | 设置边界线颜色（设为 none 时不显示边界，设为 flat 时用相应顶点的颜色单色显示边界，设为 interp 时两端顶点的颜色的插值过渡颜色显示边界） |
| FaceColor | 表示颜色的数组或字符，或 none，flat，interp | 设置多边形内部颜色（设为 none 时不显示多边形面，设为 flat 时单色显示，设为 interp 时用插值过渡色显示） |

以 patch('Vertices',vmatrix,'Faces',fmatrix,'FaceVertexCData',cmatrix,…)创建的多个片块模型为例，当 Vertices 取值为 $k$*3 数组（总共有 $k$ 个顶点）、Faces 取值为 $m$*$n$（总共有 $m$ 个片块多边形，每个片块多边形有 $n$ 个顶点）时，FaceVertexCData 属性的设置值 cmatrix 的解释方式有如下几种情况（CData 属性类似）。

（1）cmatrix 为 1 行 1 列的标量，则将所有顶点颜色都设置为该标量在当前图形的颜色表中对应的颜色，并按照 EdgeColor 和 FaceColor 的属性设置，确定边界线和区域填充色（此时这两个属性都不能设置为'interp'）。

（2）cmatrix 为 1 行 3 列的数组时，则将所有顶点颜色都设置为该一维数组代表的 RGB 色彩空间中的颜色，并按照 EdgeColor 和 FaceColor 的属性，设置确定边界线和区域填充色（此时这两个属性都不能设置为'interp'）。

（3）cmatrix 为 $k$ 行 1 列的数组时，则将 $k$ 个顶点颜色分别设置为这 $k$ 个标量在当前图形的颜色表中对应的颜色，并按照 EdgeColor 和 FaceColor 的属性设置，确定边界线和区域填充色用单色显示还是插值过渡色显示。

（4）cmatrix 为 $k$ 行 3 列的数组时，则将 $k$ 个顶点颜色分别设置为这 $k$ 个一维数组代表的 RGB 色彩空间中的颜色，并按照 EdgeColor 和 FaceColor 的属性设置，确定边界线和区域填充色用单色显示还是插值过渡色显示。

（5）cmatrix 为 $n$ 行 1 列的数组时，则将 $n$ 个多边形区域内填充色设为这 $n$ 个标量在当前图形的颜色表中对应的颜色，即单色显示，此时 FaceColor 的属性不能设置为'interp'。

（6）cmatrix 为 $n$ 行 3 列的数组时，则将 $n$ 个多边形区域内填充色设为这 $n$ 个一维数组代表的 RGB 色彩空间中的颜色，即单色显示，此时 FaceColor 的属性不能设置为'interp'。

关于片块模型的顶点、边界线、区域颜色的设置，MATLAB 帮助文件中有更详细的说明，有兴趣的读者可以自行参考。





**例 24-16**　设置多个片块模型的颜色。

**解：** 在命令窗口中输入：

```
%Ex24-16 set colors in multi-patches model
clear
close all
Vm=[0 0 0;1 0 0;1 0 1;0 0 1;0 1 1;1 1 1;1 1 0];
Fm=[1 2 3 4;3 4 5 6;6 3 2 7];
subplot(3,2,1)
patch('Vertices',Vm,'Faces',Fm,'FaceVertexCData',rand(1,1),'FaceColor','flat')
view([30,30]);title('set uniform index-color')
subplot(3,2,2)
patch('Vertices',Vm,'Faces',Fm,'FaceVertexCData',rand(1,3),'FaceColor','flat')
view([30,30]);title('set uniform RGB-color')
subplot(3,2,3)
patch('Vertices',Vm,'Faces',Fm,'FaceVertexCData',rand(size(Vm,1),1),'FaceColor','interp')
view([30,30]);title('assign multi-index-color to Vertices')
subplot(3,2,4)
patch('Vertices',Vm,'Faces',Fm,'FaceVertexCData',rand(size(Vm,1),3),'FaceColor','interp')
view([30,30]);title('assign multi-RGB-color to Vertices')
subplot(3,2,5)
patch('Vertices',Vm,'Faces',Fm,'FaceVertexCData',rand(size(Fm,1),1),'FaceColor','flat')
view([30,30]);title('assign multi-index-color to Faces')
subplot(3,2,6)
patch('Vertices',Vm,'Faces',Fm,'FaceVertexCData',rand(size(Fm,1),3),'FaceColor','flat')
view([30,30]);title('assign multi-RGB-color to Faces')
```

代码运行结果如图 24-17 所示。

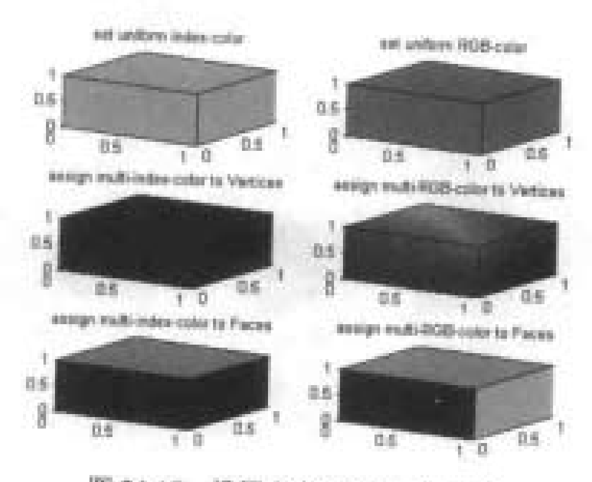



图 24-17　设置多个片块模型的颜色

# 24.3 三维图形显示控制

## 24.3.1 设置坐标轴

三维图形下坐标轴的设置和二维图形下类似，都是通过带参数的 axis 命令设置坐标轴显示范围和显示比例。

（1）axis([xmin xmax ymin ymax zmin zmax])设置三维图形的显示范围，数组元素列表确定了每一坐标轴显示的最大最小值。

（2）axis auto 则根据 $x$，$y$，$z$ 的范围自动确定坐标轴的显示范围。

（3）axis manual 锁定当前坐标轴的显示范围，除非手动进行修改。

（4）axis tight 设置坐标轴显示范围为数据所在范围。

（5）axis equal 设置各坐标轴的单位刻度长度等长显示。

（6）axis square 将当前坐标范围显示在正方形（或正方体）内。

（7）axis vis3d 锁定坐标轴比例，不随三维图形的旋转而改变。

**例 24-17** 设置坐标轴。

**解**：在命令窗口中输入：

```
%Ex24-17 set axis
close all
subplot(1,3,1)
ezsurf(@(t,s)(sin(t).*cos(s)),@(t,s)(sin(t).*sin(s)),@(t,s)cos(t),[0,2*pi,0,2*pi])
axis auto;title('auto')
subplot(1,3,2)
ezsurf(@(t,s)(sin(t).*cos(s)),@(t,s)(sin(t).*sin(s)),@(t,s)cos(t),[0,2*pi,0,2*pi])
axis equal;title('equal')
subplot(1,3,3)
ezsurf(@(t,s)(sin(t).*cos(s)),@(t,s)(sin(t).*sin(s)),@(t,s)cos(t),[0,2*pi,0,2*pi])
axis square;title('square')
```

代码运行结果如图 24-18 所示。

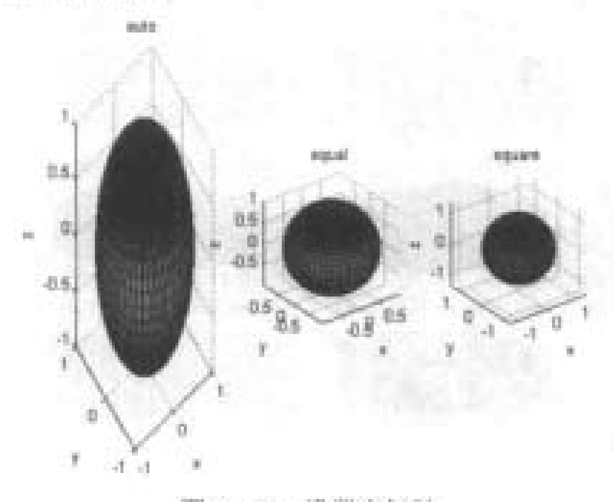



图 24-18 设置坐标轴



## 24.3.2　设置视角

在不同位置查看三维图形会看到不同的侧面和结果，因此，设置一个能够查看整个图形最主要的特性的视角，在三维图形的查看中是重要的。

MATLAB 下可以通过函数命令或图形旋转工具改变视角。旋转工具在本章 24.3.4 中介绍，这里介绍通过 view 在命令行方式下设置图形视角的方法。

view 函数的常用语法格式如表 24-6。

表 24-6　view 函数设置视角的语法格式

| 函数语法格式 | 说　明 |
|---|---|
| view(az,el)<br>view([az,el]) | 设置视角位置在 azimuth 角度和 elevation 角度确定的射线上 |
| view([x,y,z]) | 设置视角位置在[x,y,z]向量所指示的方向 |
| view(2) | 默认的二维视图视角，相当于 az = 0, el = 90 |
| view(3) | 默认的三维视图视角，相当于 az = −37.5, el = 30 |
| [az,el]=view | 返回当前视图的视角 az 和 el |

表 24-6 中参数 az 和 el 分别确定了图形中心（三维坐标轴原点）和观察者眼睛连线确定的向量与−y 轴方向和 x-y 平面的夹角，其意义如图 24-19 所示。

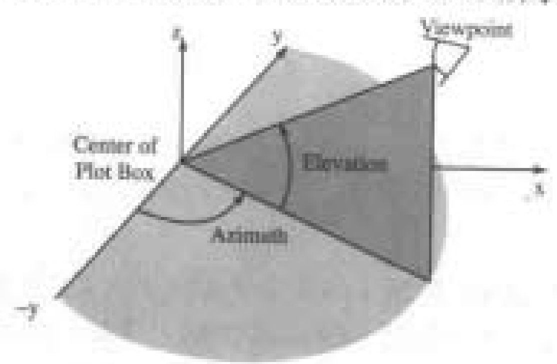



图 24-19　视角设置参数意义图示（来自 MATLAB 联机帮助）

例 24-18　设置视角。

**解：** 在命令窗口中输入：

```
%Ex24-18 set the viewpoint
clear
close all
subplot(2,2,1)
ezmesh(@peaks);
view(3);
[a,b]=view;title(mat2str([a,b]))
subplot(2,2,2)
ezmesh(@peaks);
view(2);
[a,b]=view;title(mat2str([a,b]))
subplot(2,2,3)
```

```
ezmesh(@peaks);
view([30 45]);
[a,b]=view;title(mat2str([a,b]))
subplot(2,2,4)
ezmesh(@peaks);
view([1 1 sqrt(2)]);
[a,b]=view;title(mat2str([a,b]))
```

代码运行结果如图 24-20 所示。

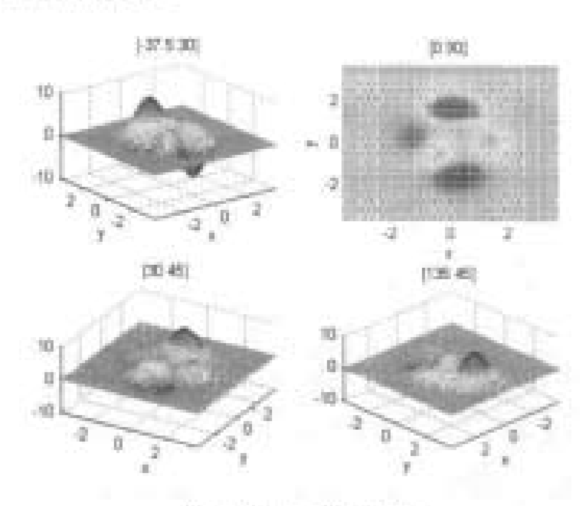


图 24-20　设置视角

### 24.3.3　Camera 控制

在 MATLAB 图形窗口下查看一幅三维图形，类似于用户的眼睛作为摄像头对图形场景进行拍摄。MATLAB 基于这一类比，提供了 Camera 控制工具条，可供用户便捷地调节图形查看效果。

默认的图形窗口下，Camera 控制工具条是不显示的，选择 View 菜单下的 Camera Toolbar，可以在当前窗口显示（或隐藏）Camera 控制工具条，如图 24-21 所示。

图 24-21　Camera 控制工具条

图 24-21 的工具条中第一组工具按钮是控制 Camera（用户眼睛）和图形相对位置的，从左向右依次是：

（1）Camera 圆周旋转按钮，固定图形位置，用户眼睛在到坐标轴原点的圆周上旋转查看；

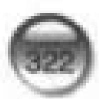
（2）场景灯光旋转按钮，设置光源相对于坐标原点和用户眼睛连线的角度；

（3）图形圆周旋转按钮，用户固定眼睛，图形（以坐标轴原点为准）在以用户眼睛为圆心的圆周上旋转时，用户查看图形的效果；

（4）Camera 平移按钮，固定图形位置，用户眼睛水平或垂直移动；

（5）Camera 推进或后退按钮，不改变视角的情况下，改变用户眼睛和图形之间的距离；

（6）Camera 缩放，增大或缩小用户眼睛观察时取景的角度；



（7）Camera 旋转，用户眼睛和图形位置固定，绕连线轴旋转眼睛观察。

紧邻的第二组工具按钮用来设置当前图形坐标轴取向；第三组工具按钮设置当前图形场景光源；第四组工具按钮设置透视模式；最后一组工具按钮用来重置或终止 Camera 移动和场景灯光。

这些工具按钮的应用效果要通过实际的例子体会，用户可以针对自己的三维图形窗口进行练习。另外，MATLAB 帮助文件中也有形象地解释这些按钮效果的图片，感兴趣的读者请自行参考。

### 24.3.4 其他控制工具

三维图形窗口下还有其他的图形控制工具，包括缩放、平移和旋转等，其操作和 Camera 工具按钮类似，即选择相应的工具按钮后，在绘图时拖拽鼠标来实现按钮功能，这些操作都很简单，用户通过自己的三维图形实际练习即可。

三维图形的标注也和二维图形一样，用户可以通过多种方式在三维图形上添加各种标注对象，本书第 23 章已经详细讲述，此处不再赘述。

另外需要补充说明的是，颜色条标注在二维图形下基本不使用，但在三维图形下，尤其是使用了索引颜色表的三维图形，标注出颜色条可以让用户清楚地理解图形中颜色所代表的数值范围，因此，颜色条标注在三维图形下是很常用的。

三维图形还有很多效果设置，包括颜色、光照等，这些将在本书第 25 章中介绍。

## 24.4 小结

本章讲述了 MATLAB 中三维数据可视化方法，这包括基本的三维曲线图和三维曲面图的绘制，三维片块模型的创建、设置和三维图形显示设置。

其中，基本的三维图形的绘制和显示设置是本章的重点，尤其是网格点的概念和各种三维图形的区别，用户应该仔细体会和理解。另外，三维图形中颜色表和数值对应这一概念也是三维数据可视化中重要的概念，本书第 25 章还会讲解，用户可以通过本章例子学习。显示设置部分，视角设置是三维图形查看中最重要的设置项，view 命令的各种语法形式读者应该做到能够熟练应用。

本章中三维模型的部分，在创建实体模型中会用到，对一般读者来说，作为提高兴趣的参考内容即可。

# 第 25 章 使用颜色和光影

在数据可视化技术中，用户不但可以通过二维或三维图形显示个体数据，而且可以通过设置颜色，增加一个维度的信息显示方式，或者通过设置丰富的颜色变化效果、光影效果使图形显示更加美观，这就是本章要介绍的内容——MATLAB 中颜色和光影效果的设置。

## 25.1 MATLAB 中的颜色

### 25.1.1 着色技术

MATLAB 在图形显示中，可以对数据点进行着色，从而使图形更加美观生动。在对数据点着色时，MATLAB 有真彩色着色和索引着色两种处理方法。

（1）RGB 真彩色着色是采用 RGB 颜色空间，对每一个数据点都需要指定一个 RGB 三元数组，这一 RGB 三元数组为此数据点确定了 RGB 颜色空间中的一种特定颜色。

（2）索引着色则使用了 MATLAB 图形窗口的颜色表，颜色表是一个 $m$*3 的数组，每一行实际上构成了一个 RGB 三元数组，从而确定了一种颜色。在对数据点着色时，以直接索引或映射索引的方式把数据点的 $z$ 值转换为颜色表索引（即行标），从而确定此数据点颜色为颜色表中该行指定的颜色。

在实际使用中，通常有下列三种处理方式：

（1）当用户绘图不指定数据点的颜色时，MATLAB 采用默认的颜色表，而且把数据点 $z$ 值范围映射为颜色表索引范围，然后通过映射索引的方式由数据点 $z$ 值确定每一个数据点对应于颜色表中的索引和相应的颜色；

（2）当绘图中有 $m$*$n$ 个数据点，并且用户指定使用了一个 $m$*$n$ 的颜色索引表时，MATLAB 则通过直接索引的方式，将此 $m$*$n$ 个数据点颜色设置为其索引值对应于颜色表中



的色彩；

（3）当绘图中有 $m*n$ 个数据点，并且用户指定使用了一个 $m*n*3$ 的颜色数组时，MATLAB 则使用 RGB 真彩着色方式，将每一个数据点颜色设置为对应的颜色数组指定的 RGB 颜色。

## 25.1.2 RGB 真彩着色

RGB 真彩着色需要计算机显示器支持 24 位真彩显示，这时候显示器可显示颜色有 $2^{24}$ 种（超过 1600 万）。

RGB 色彩是通过一个 RGB 三元数组[*r g b*]指定的，这三个数字分别称为色彩的 R、G、B 分量，其取值范围应在 0 到 1 之间。

常用的某些 RGB 色彩空间的色彩的 R，G，B 分量如表 25-1 所示。

表 25-1 常用色彩的 R、G、B 分量

| 颜 色 | R | G | B |
|---|---|---|---|
| 黑（Black） | 0 | 0 | 0 |
| 白（White） | 1 | 1 | 1 |
| 红（Red） | 1 | 0 | 0 |
| 绿（Green） | 0 | 1 | 0 |
| 蓝（Blue） | 0 | 0 | 1 |
| 黄（Yellow） | 1 | 1 | 0 |
| 洋红（Magenta） | 1 | 0 | 1 |
| 青（Cyan） | 0 | 1 | 1 |
| 灰（Gray） | 0.5 | 0.5 | 0.5 |
| 暗红（Dark red） | 0.5 | 0 | 0 |
| 铜（Copper） | 1 | 0.62 | 0.40 |
| 碧绿（Aquamarine） | 0.49 | 1 | 0.83 |

在图形绘制时，通过指定一个 $m*n*3$ 的三维数组表示数据点的颜色，就可以实现 RGB 真彩着色，这个数组的每一页分别存储对应的 $m*n$ 个数据点颜色的 R，G，B 分量。

**例 25-1** RGB 真彩着色。

**解**：在命令窗口中输入：

```
%Ex25-1 RGB coloring
clear
close all
[x,y]=meshgrid(-1:0.2:1);
z=x.^2+cos(y*pi);
[m,n]=size(z);
c=rand(m,n,3);
surf(x,y,z,c)
title('RGB coloring of surf')
```

代码运行结果如图 25-1 所示。

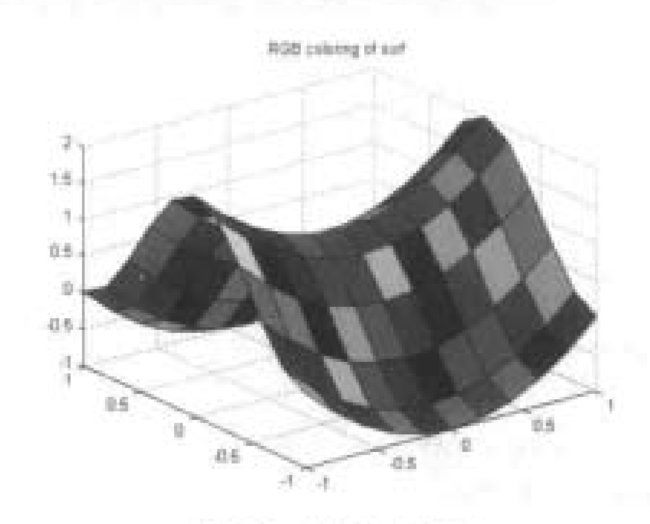



图 25-1　RGB 真彩着色

这时候用 whos 查看 MATLAB 工作区变量的结果如下。

```
>> whos
  Name      Size                          Bytes  Class

  c         11x11x3                        2904  double array
  m          1x1                              8  double array
  n          1x1                              8  double array
  x         11x11                           968  double array
  y         11x11                           968  double array
  z         11x11                           968  double array

Grand total is 728 elements using 5824 bytes
```

由此可见，对于 11*11 的绘图数据点，通过 rand(*m*,*n*,3)生成了了一个 11*11*3 的数值数组，其元素范围都在 0 到 1 之间，这个数组确定了对应的 11*11 个数据点的颜色，然后 surf 函数在绘制表面图时就将相应的数据点设置成对应的 RGB 颜色。

MATLAB 中支持真彩着色的着色器只有 OpenGL 和 Z-buffer 两个，一般情况下 MATLAB 图形窗口的着色器模式（RendererMode）属性都是设置为 auto，这时候 MATLAB 自动选择 Z-buffer 进行真彩着色。

当手工选择了着色器（Renderer）为 painters 时，用户在绘制 RGB 真彩着色的表面图、片块模型和图像时，MATLAB 会返回警告并且无法完成着色操作。

## 25.1.3　颜色表

对于 *m***n* 个数据点的图形用 RGB 着色，需要指定一个 *m***n**3 的颜色数组，当数据点规模很大时，这个颜色数组也将会非常庞大，为了减少不必要的内存开销，这时候可以用 MATLAB 提供的另一种着色技术——索引着色。

索引着色需要启用颜色表。对于每一个 MATLAB 图形窗口都有惟一的颜色表。颜色表实际上是一个 *m**3 的数值数组，其元素取值也在 0 到 1 之间，每一行元素实际上组成了一





个 RGB 三元数组，确定一个 RGB 色彩空间的特定颜色。因此，$m$*3 的颜色表的 $m$ 行数据就确定了 $m$ 种颜色，这构成了索引着色图的颜色空间。

MATLAB 内置了许多颜色表，如 jet、hot 等，图 25-2 显示了这些内置颜色表的颜色变化范围和名称。

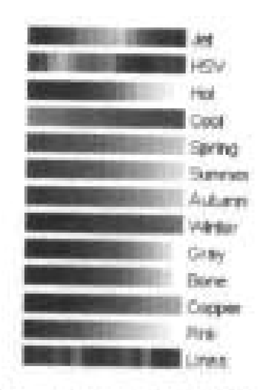



图 25-2 MATLAB 内置的颜色表

可以通过类似 colormap(name)这样的语句，将当前图形窗口的颜色表设置为 name 命名的 MATLAB 内置颜色表。colorbar 可以控制当前图形的颜色表标注的显示。用户也可以以这些内置颜色表为基础，生成颜色种类较少的缩减颜色表，如 hot(8)就可以以 hot 颜色表为基础，生成只有 8 种颜色的颜色表数组。

由于颜色表实际上是一个数值数组，如 MATLAB 内置的颜色表都是 64*3 的数组（即包括了 64 种颜色），因此，用户可以用自定义创建 $m$*3 的颜色数组来设置当前图形窗口的颜色表。

colormap(cmap)函数将当前图形窗口的颜色表设为 cmap 数组表示的颜色表，需要注意的是，cmap 数组必须是一个 m*3 的数值数组，其所有元素取值都在 0 到 1 之间，否则设置颜色表时会出错，$m$ 值代表当前图形中索引色彩的种类。

绘制索引着色的图形的第一步，就是通过 colormap 设置当前图形窗口的颜色表。当用户不指定时，MATLAB 默认将 jet 颜色表设置为当前窗口的颜色表。

**例 25-2** 颜色表数组操作。

**解**：在命令窗口中输入：

```
>> A=cool;  %返回 cool 颜色表数组的值
>> size(A)
ans =
    64     3
>> A(60:64,:)    %裁减颜色表数组
ans =
    0.9365    0.0635    1.0000
    0.9524    0.0476    1.0000
    0.9683    0.0317    1.0000
    0.9841    0.0159    1.0000
    1.0000         0    1.0000
>> cool(8)  %以 cool 颜色表为基础创建缩减颜色表数组
ans =
```

```
     0     1.0000    1.0000
0.1429    0.8571    1.0000
0.2857    0.7143    1.0000
0.4286    0.5714    1.0000
0.5714    0.4286    1.0000
0.7143    0.2857    1.0000
0.8571    0.1429    1.0000
1.0000         0    1.0000
```

### 25.1.4 索引着色

**1．映射索引着色**

设定图形窗口的颜色表后，用户就可以在绘图指令中指定绘图数据点的颜色索引数组，实现索引着色绘图了。MATLAB 会将此颜色索引数组的数值映射到颜色表索引（行号）值范围，然后通过映射索引的方式，确定颜色索引数组中对应位置上的数据点的索引颜色。

例 25-3 指定颜色索引数组的映射索引着色。

**解**：在命令窗口中输入：

```
%Ex25-3 Scale-Mapping Indexed Coloring (With color matrix)
clear
close all
[x,y]=meshgrid(-1:0.1:1);
z=x.*y+sin(x.*y.*pi);
c=randn(size(z));
surf(x,y,z,c)
colormap(cool)
colorbar
```

代码运行结果如图 25-3 所示。

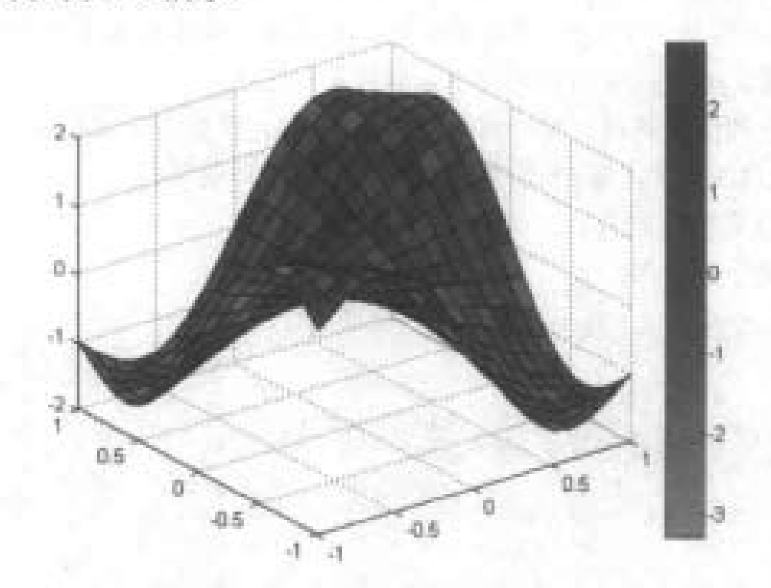
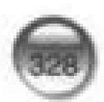



图 25-3 指定颜色索引数组的映射索引着色

在 MATLAB 工作区查询索引数组 *c* 的取值范围，代码如下：





```
>> max(max(c))
ans =
    2.8149
>> min(min(c))
ans =
   -3.2854
```

从查询结果和图 25-3 可以看出，MATLAB 将索引数组 *c* 的取值范围（[-3.2854 2.8149]）映射到颜色表的行号范围（[1 64]），*c*(*a*,*b*)的值确定数据点（x(*a*,*b*), y(*a*,*b*), z(*a*,*b*)）的颜色为颜色表中第 *k* 行的颜色，*k*=fix((*c*(*a*,*b*)-min(min(*c*)))/(max(max(*c*))-min(min(*c*)))*len_cm)+1，其中，len_cm 是颜色表的颜色种类，即 len_cm=size(colormap,1)。

当不设置颜色索引数组时，MATLAB 会默认以数据点 *z* 值为颜色索引数组，进行类似上面的映射索引操作，确定数据点的颜色。

**例 25-4**　不指定颜色索引数组的映射索引着色。

**解**：在命令窗口中输入：

```
%Ex25-4 Scale-Mapping Indexed Coloring (Without color matrix)
clear
close all
[x1,y1]=meshgrid(-1:0.1:0);
z1=zeros(size(x1));
[x2,y2]=meshgrid(0:0.1:1);
z2=ones(size(x2));
x=[x1 x2];y=[y1 y2];z=[z1 z2];
surf(x,y,z)
colormap(cool)
colorbar
```

代码运行结果如图 25-4 所示。

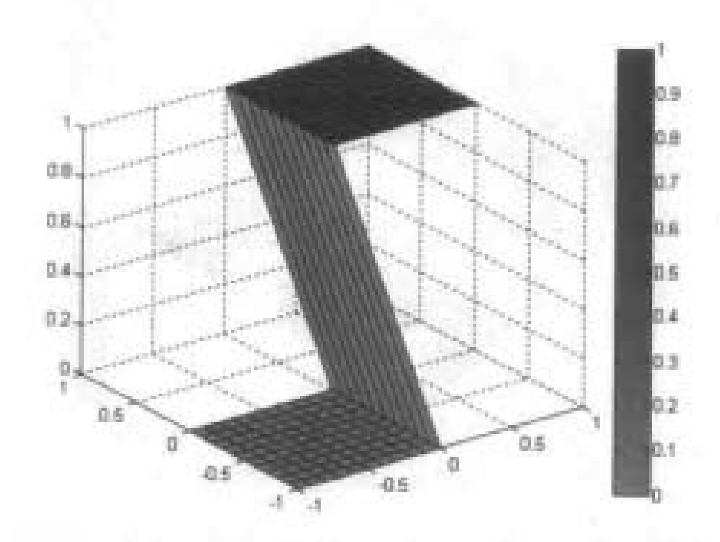



图 25-4　不指定颜色索引数组的映射索引着色

从例 25-4 和图 25-4 可以看到，在不指定颜色索引数组的情况下，MATLAB 自动把数据点 *z* 值范围（[0 1]）映射到颜色表行号范围（[1 64]），对于 *z*=0 的区域用颜色表第一行

表示的颜色着色，对 $z$=1 的区域用颜色表最后一行表示的颜色着色。

2. 直接索引着色

MATLAB 中确定数据点颜色在颜色表中的索引时，也可以通过直接索引方式，即颜色索引数组取值就是相应数据点颜色在颜色表中的行号，如果索引值不是整数，则将其向零取整后的结果作为索引值。当此数值在颜色表的行号范围内（如[1 64]），则相应点颜色设为颜色表中对应行的颜色，若数值小于 1，则相应点颜色设为颜色表中第一行的颜色，若数值大于颜色表最大行号，则相应点颜色设为颜色表中最后一行的颜色。

由于默认情况下，MATLAB 是采用映射索引方式实现索引着色的，所以绘图时需要手工设置'CdataMapping'属性值为'direct'，这样实际绘图才会采用直接索引方式。

**例 25-5** 直接索引着色。

**解**：在命令窗口中输入：

```
%Ex25-5 Direct-Mapping Indexed Coloring
clear
close all
[x,y]=meshgrid(0:2);
z=x+y.^2;
surf(x,y,z,[-50  0  15;25  35  45;55  65  85],'CDataMapping','direct','
FaceColor','interp')
colorbar
```

例 25-5 代码中设置'FaceColor'属性为'interp'可以使表面图显示为插值颜色过渡模式。

运行代码后，并手动添加数据点标记，得到如图 25-5 所示的结果。

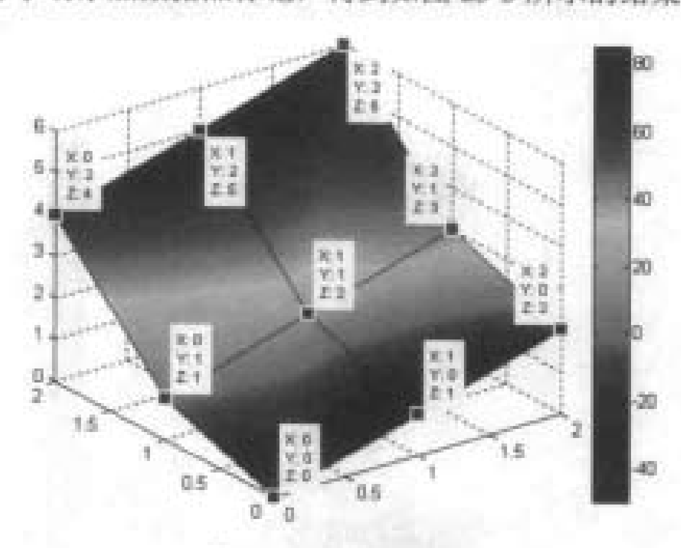


图 25-5 直接索引着色

从图 25-5 结果中可以看到，索引数值为-50，0 这些小于 1 的数值时，该点颜色设置为颜色表中第一行颜色（暗蓝色），当索引数值为 15，25，35，45，55 这些介于 1 到 64 的数字时，相应点颜色设置为颜色表中对应行的颜色（分别为亮蓝、淡蓝、黄绿、棕黄、暗红等），当索引值为 65，85 这些大于 64 的数字时，相应点颜色设置为颜色表中最后一行的颜色（暗红）。



### 25.1.5　shading 模式

指定了数据点的颜色后，MATLAB 在显示三维曲面图、片块模型、图像时，还可以通过带参数的 shading 命令设置 shading 模式。

（1）shading flat 将图形对象（如网线图中的线条、表面图中的表面区域和片块模型中的片块多边形等）显示为单色。

（2）shading faceted 则将区域图形对象显示为单色，而将线条对象显示为黑色，这也是 MATLAB 中默认的 shading 模式。

（3）shading interp 则将图形对象显示为颜色过渡模式。

**例 25-6**　shading 模式。

**解**：在命令窗口中输入：

```
%Ex25-6 Shading mode
clear
close all
[x,y]=meshgrid(-3:0.5:3);
z=x.*y+sin(x*pi)+cos(y*pi);
subplot(2,2,1)
surf(x,y,z)
title('no shading')
subplot(2,2,2)
surf(x,y,z)
shading flat
title('shading flat')
subplot(2,2,3)
surf(x,y,z)
shading faceted
title('shading faceted')
subplot(2,2,4)
surf(x,y,z)
shading interp
title('shading interp')
```

代码运行后，图形绘制结果如图 25-6 所示。

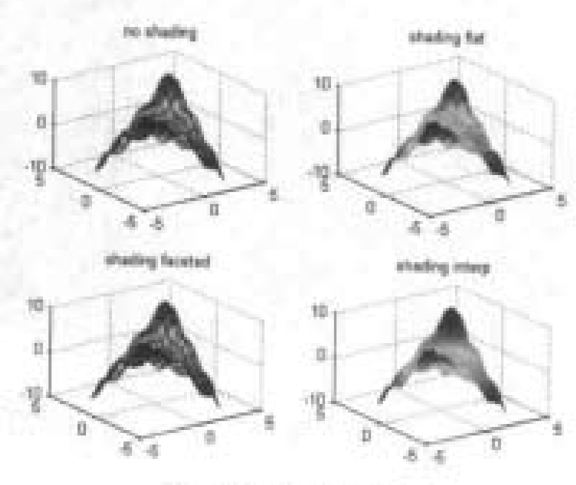



图 25-6　shading 模式

## 25.2 光照效果

除了设置颜色，MATLAB 中还可以通过设置光照效果，增强图形的美观和逼真程度。光照效果主要用在表面图和片块模型中。

### 25.2.1 光源对象

MATLAB 给图形添加光照效果，需要创建一个光源对象，这可以通过 light 命令实现，其常用的语法格式为 light('PropertyName',PropertyValue,…)，即通过设置光源对象的各种属性来创建光源对象。

光源对象常用的属性项包括：Color，Style 和 Position。

（1）Color 属性定义了光源发射光的颜色，其取值可以是 RGB 三元数组或表示颜色的字符串。

（2）Style 属性定义光源类型，取值为'infinite'时设置光源发射平行光，取值为'local'时设置光源为点光源，发射辐射状光线。MATLAB 中默认取平行光源。

（3）Position 属性对于不同类型的光源意义不同，对于平行光源 Position 定义了光线发射的方向，对于点光源 Position 定义了点光源的位置。

**例 25-7** 光源对象。

**解：**在命令窗口中输入：

```
%Ex25-7 Light Object
clear
clc
[x,y]=meshgrid(-1:0.1:1);
z=sin(x*pi)+cos(y*pi);
subplot(2,2,1)
surf(x,y,z)
title('no light')
subplot(2,2,2)
surf(x,y,z)
light('Color','r','Style','infinite','Position',[0 1 2])
title('red infinite light')
subplot(2,2,3)
surf(x,y,z)
light('Color','g','Style','infinite','Position',[0 1 2])
title('green infinite light')
subplot(2,2,4)
surf(x,y,z)
light('Color','r','Style','local','Position',[0 1 2])
title('red local light')
```

代码运行后结果如图 25-7 所示。

光源对象还有很多属性项设置，可以实现多种多样的光照效果，请读者参考 MATLAB 帮助文档学习。



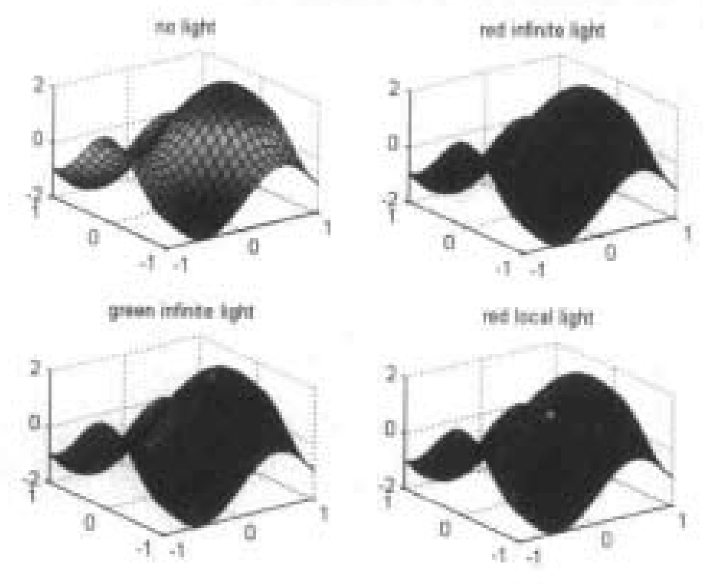



图 25-7 光源对象

## 25.2.2 光照方法

光源如何改变图形的显示结果，一个重要的因素就是光照方法的设置。光照方法设置不同，光源对与图形中边界线、表面区域颜色的影响也将不同。

MATLAB 中支持三种光照方法：flat，gouraud 和 phong。这可以通过设置光源的 FaceLighting 和 EdgeLighting 属性项的取值来改变光照方法。

通过 lighting 命令则可以便捷地设置光照方法。

（1）lighting flat 设置光照方法为 flat，这种光照方法下，光源对图形中每一个对象产生同样的效果，这比较合适于显示由多个表面区域组成的图形。

（2）lighting gouraud 设置光照方法为 gouraud，这种光照方法下，MATLAB 计算光照影响下每一个顶点的颜色，然后插值确定区域内颜色，这适合于显示弯曲的表面图形。

（3）lighting phong 设置光照方法为 phong，这种光照方法和 gouraud 类似，但 MATLAB 还会附加计算每一个图形像素点的反射系数，一般来说 phong 方法比 gouraud 方法显示效果更逼真生动，但耗时更久。

（4）lighting none 关闭光照效果。

**例 25-8** 光照方法。

**解**：在命令窗口中输入：

```
%Ex25-8 Lighting Method
clear
close all
[x,y]=meshgrid(-1:0.2:1);
z=sin(x*pi)+cos(y*pi);
subplot(2,2,1)
surf(x,y,z)
light('Color','r','Style','infinite','Position',[1 -1 2])
```

```
lighting none
title('lighting none')
subplot(2,2,2)
surf(x,y,z)
light('Color','r','Style','infinite','Position',[1 -1 2])
lighting flat
title('lighting flat')
subplot(2,2,3)
surf(x,y,z)
light('Color','r','Style','infinite','Position',[1 -1 2])
lighting gouraud
title('lighting gouraud')
subplot(2,2,4)
surf(x,y,z)
light('Color','r','Style','infinite','Position',[1 -1 2])
lighting phong
title('lighting phong')
```

代码运行后，绘图结果如图 25-8 所示。

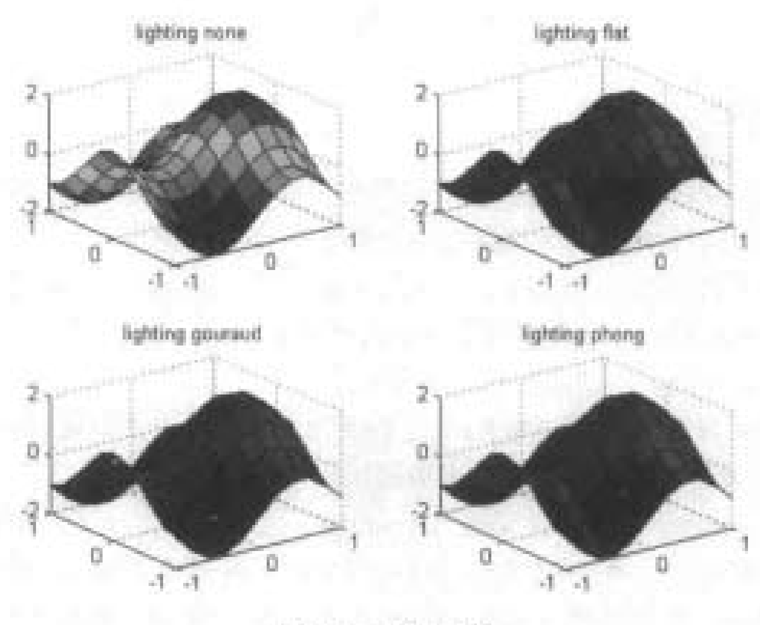



图 25-8 光照方法

## 25.3 小结

本章介绍了 MATLAB 中丰富数据可视化结果的两种方法——颜色和光照。其中详细讲解了 MATLAB 中两种着色技术，尤其是索引着色技术的原理，读者应该仔细理解，因为 MATLAB 中许多对象的颜色设置都是类似的。其他部分的内容，读者可以根据兴趣选学，特别感兴趣的读者，还可以参考 MATLAB 中关于颜色、光照以及透明效果的联机帮助。

# 第 26 章

# 图像、声音和视频

本章介绍 MATLAB 中处理图像、声音和视频文件的方法。

## 26.1 图像

### 26.1.1 图像及其数值类型

MATLAB 能够读入、显示和处理多种标准图像格式文件。当图像文件经由 MATLAB 读入后，都是用数值数组的方式来表示的，这时候通过 MATLAB 函数显示出来的图像，实际上是一个句柄图形的图像对象。MATLAB 支持的标准图像格式包括：BMP，HDF，JPEG，PCX，PNG，TIFF，XWD 等。

MATLAB 读入这些标准图像文件后，用数值数组存储这些文件的信息。其中，数值数组的数值元素可以是无符号的 8 位整数、无符号的 16 位整数或双精度浮点数。通常情况下，一个标准格式的图像文件经由 MATLAB 读入后产生至少一个数组，称为数据数组，其中记录了标准格式的图像中各像素点的颜色信息；大多数情况下还会产生第二个数组，称为颜色表数组，用来存储解释数据数组的数值对应的颜色信息。

根据 MATLAB 读入图像后产生的数组个数和 MATLAB 再次显示、处理这些数组数据时对数值的解释方法，可以把 MATLAB 内部的图像对象分为索引图像，灰阶强度图像和 RGB 真彩图像这三种类型。

（1）索引图像

索引图像包含两个数值数组：一个是用来储存各像素点颜色索引的二维数据数组 X，另一个是颜色表数组 map。

数据数组 *X* 有 *m* 行 *n* 列个元素，记录了 *m* 行 *n* 列个像素点的颜色索引，其元素可以是双精度浮点数、无符号 8 位或 16 位整数中的任何一种类型。颜色表数组 map 是一个双精度浮点类型的数值数组，包含多行 3 列，每一行的三个元素共同指定了 RGB 色彩模式下的一种颜色。当数据数组 *X* 为双精度浮点类型时，*X* 某行某列上如果取值为 *N*，则代表该行该列的像素点颜色为颜色表数组 map 中第 N 行数字指定的某种 RGB 色彩空间的颜色；当数据数组 *X* 为无符号整数类型时（8 位或 16 位），*X* 某行某列上如果取值为 *N*，则代表该行该列的像素点颜色为颜色表数组 map 中第 *N*+1 行数字指定的某种 RGB 色彩空间的颜色。

（2）灰阶强度图像

灰阶强度图像只包含一个二维数值数组 *X*，用来存储各像素点的灰阶强度值，其数值可以是双精度浮点数、无符号 8 位或 16 位整数中的任何一种类型。MATLAB 显示和处理灰阶强度图像时，需要指定一个颜色表。实际上，MATLAB 是按照处理索引图像类似的方式处理灰阶强度图的。

（3）RGB 真彩图像

RGB 真彩图像也只包含一个数值数组 X，但它是一个 *m***n**3 的三维数组。这个三维数组的每一页上分别存储各个对应位置上像素点的 R、G、B 值。这个三维数组的元素也可以是双精度浮点数、无符号 8 位或 16 位整数中的任何一种类型。

### 26.1.2 图像处理函数

MATLAB 中图像相关函数如表 26-1 所示。

表 26-1　图像处理函数

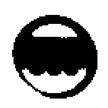
| 函　　数 | 说　　明 |
|---|---|
| imfinfo | 获取图像文件信息 |
| imformats | 获取 MATLAB 可读入的标准图像格式类型及其说明 |
| imread | 将图像格式文件读入为 MATLAB 图像对象数组数据 |
| image | 显示图像 |
| imagesc | 自动缩放数值范围显示图像 |
| colormap | 设置颜色表 |
| axis | 设置显示图像的坐标轴 |
| imwrite | 将图像对象数据写回图像格式文件 |
| exifread | 获取.jpg、.tiff 格式图像文件的 EXIF 信息 |

下面通过几个例子分别讲述这些命令的用途。

**1．获取信息命令**

表 26-1 中，获取信息的命令包括 imfinfo，imformats 和 exifread。实际上对于各种标准图像格式，MATLAB 都有专门的信息获取命令。

下面仅示例介绍 imfinfo 和 imformats 命令。





**例 26-1**　获取信息命令。

```
>> imformats

EXT       ISA     INFO        READ      WRITE     ALPHA  DESCRIPTION
------------------------------------------------------------------------------------
------------------------
bmp      isbmp   imbmpinfo   readbmp   writebmp  0     Windows Bitmap (BMP)
cur      iscur   imcurinfo   readcur          1   Windows Cursor resources (CUR)
fts fits  isfits  imfitsinfo  readfits              0         Flexible Image
Transport System (FITS)
gif      isgif   imgifinfo   readgif   writegif  0     Graphics Interchange
Format (GIF)
hdf      ishdf   imhdfinfo   readhdf   writehdf  0      Hierarchical Data
Format (HDF)
ico      isico   imicoinfo   readico           1   Windows Icon resources (ICO)
jpg jpeg isjpg imjpginfo  readjpg   writejpg  0      Joint Photographic
Experts Group (JPEG)
pbm      ispbm   impnminfo   readpnm   writepnm  0     Portable Bitmap (PBM)
pcx      ispcx   impcxinfo   readpcx   writepcx  0    Windows Paintbrush (PCX)
pgm      ispgm   impnminfo   readpnm   writepnm  0    Portable Graymap (PGM)
png     ispng impnginfo  readpng   writepng 1     Portable Network Graphics
(PNG)
pnm      ispnm   impnminfo   readpnm   writepnm  0     Portable Any Map (PNM)
ppm      isppm   impnminfo   readpnm   writepnm  0     Portable Pixmap (PPM)
ras      isras   imrasinfo   readras   writeras  1     Sun Raster (RAS)
tif tiff  istif  imtifinfo   readtif   writetif  0      Tagged Image File
Format (TIFF)
xwd      isxwd   imxwdinfo   readxwd   writexwd  0     X Window Dump (XWD)

>> imfinfo('leaf.jpg')
ans =
          Filename: 'leaf.jpg'
       FileModDate: '03-Oct-2004 01:43:44'
          FileSize: 120792
            Format: 'jpg'
     FormatVersion: ''
             Width: 1024
            Height: 768
          BitDepth: 24
         ColorType: 'truecolor'
   FormatSignature: ''
   NumberOfSamples: 3
      CodingMethod: 'Huffman'
     CodingProcess: 'Sequential'
           Comment: {}
```

例 26-1 中，imformats 列出了 MATLAB 中 imfinfo，imread，imwrite 命令支持的标准图像文件类型，从返回结果可以看到，informats 列出了各种标准图像格式、测试函数、信息获取函数、读入函数、写回函数、是否支持 alfa 通道以及图像格式描述。而 imfinfo 可以返回某个标准格式图像文件的特定信息，包括文件名、创建时间、文件尺寸、文件格式、色深、色彩模式等。

### 2．图像读入和显示命令

通常情况下，MATLAB 通过 imread 命令读入标准格式图像文件，得到描述图像的数值数组，然后通过 image、imagesc 命令，以这些数值数组为参数，就可以在 MATLAB 图形窗口坐标轴下显示图像，通过 axis，colormap 命令可以设置坐标轴比例和色彩表。

**例 26-2** 图像读入和显示。

```
>> X=imread('leaf.jpg');
>> whos
  Name      Size                          Bytes  Class

  X         768x1024x3                  2359296  uint8 array

Grand total is 2359296 elements using 2359296 bytes

>> image(X)
```

图像显示结果如图 26-1 所示。

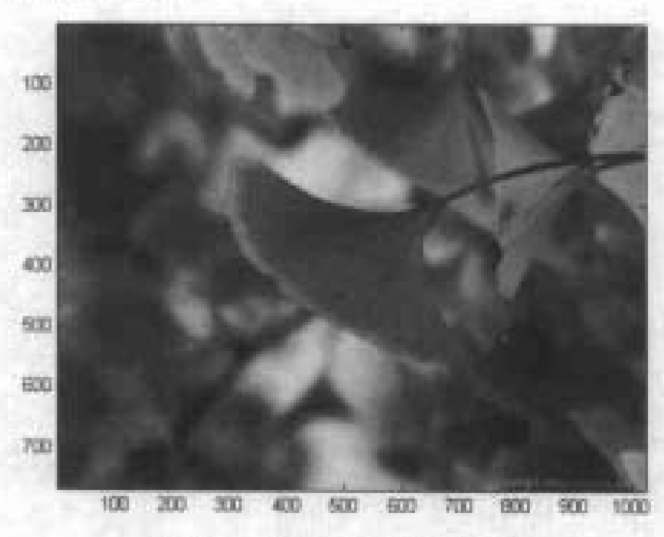



图 26-1 例 26-2 的图像显示结果

从 whos 命令结果可见，文件 leaf.jpg 经由 imread 命令读入后，产生了一个数值数组 *X*，它是一个 768*1024*3 的三维数组，这是因为 leaf.jpg 是真彩色模式的图像文件，其宽为 768 像素，长为 1024 像素。通过 image 命令将数值数组 X 代表的图像显示后，结果显示在 MATLAB 的绘图窗口下，并标出了像素坐标位置。

RGB 真彩色图像可以直接通过 image(*X*)显示，而其他两种 MATLAB 图像类型稍有不同：

（1）image(*X*)，colormap(map)用于显示数据数组为 *X*，颜色表数组为 map 的索引图像；

（2）imagesc(*X*,[0 1])，colormap(gray)用来显示数据数组为 *X* 的灰阶强度图像，设置其颜色表为 gray（结果为灰度图），当灰阶强度图像的数据数组 *X* 的数据范围并不是[0 1]时，可以直接通过 imagesc(*X*); colormap(gray)来显示，这相当于 imagesc(*X*,[min(*X*(:)) max(*X*(:))]); colormap(gray)，即把数值数组 *X* 中的最小数值映射为 gray 颜色表中 0 代表的颜色（黑），



把数值数组中的最大数值映射为 gray 颜色表中 1 代表的颜色（白）。当然，也可以通过 colormap 设置图像显示时采用其他颜色表。

**例 26-3** 灰阶强度图像显示——gray 颜色表。

```
>> figure
>> subplot(2,2,1)
>> A=ones(100,50);
>> imagesc(A,[0 1]);colormap(gray)
>> title('1 在 gray 颜色表中代表白色')
>> subplot(2,2,2)
>> B=zeros(100,50);
>> imagesc(B,[0 1]);colormap(gray)
>> title('0 在 gray 颜色表中代表黑色')
>> subplot(2,1,2)
>> C=randn(200,50);
>> imagesc(C);colormap(gray)
>> title('正态分布的二维图像显示')
```

绘图和图像显示结果如图 26-2 所示。

**例 26-3（续）** 灰阶强度图像显示——cool 颜色表。

```
>> figure
>> D=rand(300,300);
>> imagesc(D,[0,1]);colormap(cool)
```

图像显示结果如图 26-3 所示。

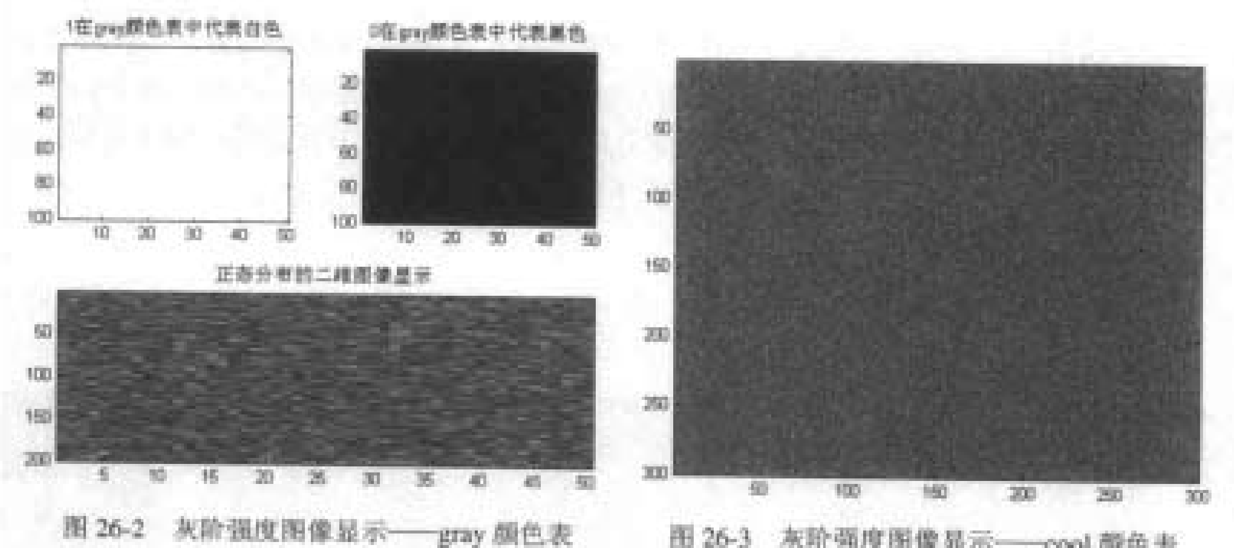



图 26-2 灰阶强度图像显示——gray 颜色表

图 26-3 灰阶强度图像显示——cool 颜色表

**例 26-4** 设置图像显示时的坐标轴比例。

```
>> load earth
>> whos
  Name      Size                    Bytes  Class

  X         257x250                514000  double array
  map        64x3                    1536  double array

Grand total is 64442 elements using 515536 bytes
```

```
>> subplot(1,2,1)
>> image(X);colormap(map)
>> title('自动设置坐标轴显示比例')
>> subplot(1,2,2)
>> image(X);colormap(map);axis image
>> title('设置横纵坐标轴等像素比例显示')
```

图像显示结果如图 26-4 所示。

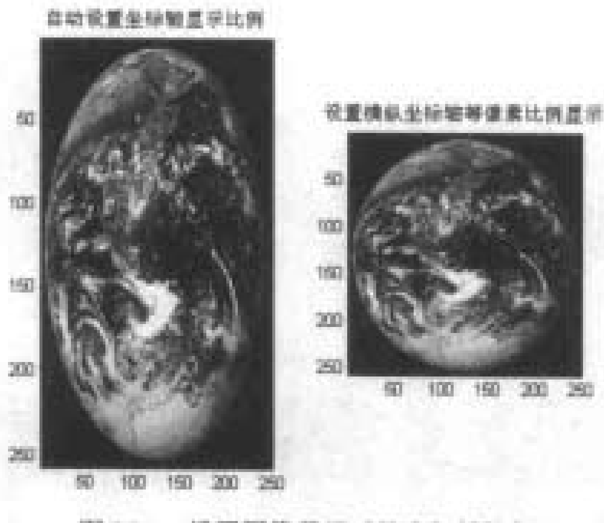



图 26-4　设置图像显示时的坐标轴比例

例 26-4 中，load earth 装载了 MATLAB 自带的例子中的图像数值数组数据，whos 显示 load 装载了数组 **X** 和数组 map。当不通过 axis 设置坐标轴显示比例时，MATLAB 会自动缩放横纵方向的显示比例，而通过 axis image 命令设置后，MATLAB 则会等比例显示横纵方向的实际像素点，因而保持了实际的图像长宽比例。

### 3. 图像写回命令

MATLAB 中可以通过 imwrite 命令把数值数组代表的数据写回为标准格式的图像文件。

**例 26-5**　图像写回命令 imwrite。

```
>> dir *.jpg

leaf.jpg

>> A=imread('leaf.jpg');
>> whos
  Name      Size                          Bytes  Class

  A         768x1024x3                    2359296  uint8 array

Grand total is 2359296 elements using 2359296 bytes

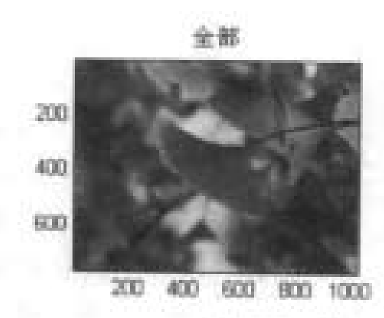
>> B=A(300:500,200:900,:);
>> imwrite(B,'leaf-part.jpg')
>> dir *.jpg
```



```
leaf-part.jpg  leaf.jpg

>> C=imread('leaf-part.jpg');
>> subplot(1,2,1)
>> image(A);axis image;title('全部')
>> subplot(1,2,2)
>> image(C);axis image;title('部分')
```

图像显示结果如图 26-5 所示。



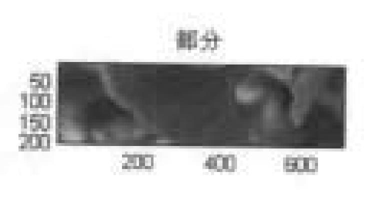




图 26-5　例 26-5 图像显示结果

从两次 dir *.jpg 命令的显示结果可以看到，imwirte 语句把从 A 裁减出来的部分数值数组 B 代表的图像信息写回到了 leaf-part.jpg 文件中了。image 显示结果也证实了这一点。

实际上，MATLAB 中还提供了很多图像处理函数，它们可以完成图像类型的转换、图像对象属性的设置等，本章只是面向 MATLAB 入门用户，因此不再作更多介绍。感兴趣的读者可以参考 MATLAB 联机帮助中和图像处理相关的内容。另外，MATLAB 图像处理工具箱是 MATLAB 中图像处理的各种功能强大函数的集合，读者也可以参考学习。

## 26.2　声音

MATLAB 能够支持 NeXT/SUN SPARCstation 声音文件（.au）、Microsoft WAVE 声音文件（.wav）、各种 Windows 兼容的声音设备、录音和播音对象，以及线性法则音频信号和 mu 法则音频信号。MATLAB 可以对声音进行读、写、获取信息、录制等操作。

表 26-2 列出了 MATLAB 操作声音的函数。

表 26-2　声音操作函数

| 函　数 | 说　明 |
|---|---|
| audioplayer | 创建一个音频播放器对象 |
| audiorecorder | 创建一个音频录制器对象 |
| mmfileinfo | 获取多媒体文件信息 |
| beep | 响铃 |
| lin2mu | 将线性法则的音频信号转换为 mu 法则的音频信号 |
| mu2lin | 将 mu 法则的音频信号转换为线性法则的音频信号 |
| sound | 将向量转换为音频信号 |

续表

| 函 数 | 说 明 |
|---|---|
| soundsc | 自动缩放将向量转换为音频信号 |
| auread | 读 NeXT/SUN SPARCstation 声音文件（.au） |
| auwrite | 写 NeXT/SUN SPARCstation 声音文件（.au） |
| aufinfo | 获取 NeXT/SUN SPARCstation 声音文件（.au）信息 |
| wavread | 读 Microsoft WAVE 声音文件（.wav） |
| wavwrite | 写 Microsoft WAVE 声音文件（.wav） |
| wavplay | 通过音频输出设备播放声音 |
| wavrecord | 通过音频输入设备录制声音 |
| wavinfo | 获取 Microsoft WAVE 声音文件（.wav）信息 |

通常，MATLAB 通过函数 sound、soundsc 将向量转换为音频信号，或者通过 auread、wavread 函数读取文件获得 MATLAB 音频信号，或者通过 wavrecord 从音频输入设备录制声音信号，然后以这些音频信号为输入参数利用 audioplayer 函数创建一个音频播放器对象，最后就可以使用方法操作音频播放器来实现声音的播放、暂停、恢复播放、终止等。

以上函数的具体调用格式和输入输出参数意义，请读者参考 MATLAB 帮助。

## 26.3 视频

MATLAB 中的视频对象称为 MATLAB movie。MATLAB 可以读入 avi 视频文件得到 MATLAB movie 数据，并对其进行写出文件或播放等操作。MATLAB 也可以把图像转换为视频帧，进而创建 MATLAB movie。

MATLAB 中对视频操作的函数，见表 26-3 所示。

表 26-3 视频操作函数

| 函 数 | 说 明 |
|---|---|
| mmfileinfo | 获取多媒体文件信息 |
| aviinfo | 获取 avi 视频文件信息 |
| aviread | 读取 avi 视频文件得到 MATLAB movie 视频帧 |
| mov2avi | 将 MATLAB movie 转换为 avi 视频文件 |
| movie | 播放 MATLAB movie |
| avifile | 创建 avi 视频文件 |
| im2frame | 将 MATLAB 图像转换为 MATLAB movie 视频帧 |
| frame2im | 将 MATLAB movie 视频帧转换为 MATLAB 图像 |
| getframe | 获取 MATLAB movie 视频帧 |
| addframe | 向 avi 视频文件中添加 MATLAB movie 视频帧 |
| close | 关闭 avi 视频文件 |

一般情况下，用户可以通过 aviread 读取 avi 视频文件，得到 MATLAB movie 视频帧，



或者通过 im2frame，getframe 等获取 MATLAB movie 视频帧，再以这些视频帧组成的数组作为输入参数，通过 movie 播放 MATLAB movie。或者用户可以通过 avifile 创建 avi 视频文件，然后通过 addframe 把前述方法得到的 MATLAB movie 视频帧添加到 avi 视频文件，添加修改完成后通过 close 命令关闭 avi 文件，也可以直接通过 mov2avi 直接把视频帧数组代表的 MATLAB movie 转换为 avi 视频文件。

## 26.4 小结

本章介绍了 MATLAB 中处理图像、声音和视频的方法，主要对 MATLAB 中的图像相关操作及概念进行了深入的讲解，对声音和视频的处理列出了相应的函数，并讲解了一般处理流程。

本章中图像处理部分是重点内容，对其中图像类型及其数值解释方法，图像的读入和显示，读者尤其要深刻理解并熟练应用。对于音频和视频处理的一般流程，读者也应该做到基本了解。其他部分的内容，用户则可以根据自己的兴趣选择性学习。本章未深入讲述的内容，尤其是关于声音、视频文件操作的函数用法及意义，用户可以参考 MATLAB 帮助进行自学。

# 第 27 章

# 图形的打印和导出

MATLAB 中用户绘制的图形，经常需要打印出来，或者导出为各种标准图像文件，供其他应用程序使用。本章介绍 MATLAB 中图形打印和导出的解决方案。

## 27.1 图形打印和导出概述

MATLAB 中用户绘制的图形，可以通过打印或导出的方式输出、保存，供其他应用程序调用。

MATLAB 中图形打印，可以直接打印到纸张，也可以打印到文件。

（1）直接打印输出是指把 MATLAB 图形数据从屏幕发送到打印机，由打印机输出到纸张打印输出。

（2）打印到文件是把 MATLAB 图形数据输出到一个 PostScript 文件，供将来打印时调用，再打印输出到纸张。

MATLAB 也支持将图形数据导出为各种标准图像文件，这样可以供其他应用程序方便地调用。图形导出可以按照各种图像文件标准导出为本地磁盘文件，也可以导出到 Windows 剪贴板，直接供其他应用程序粘贴调用。

MATLAB 中图形打印和导出，可以通过图形用户界面的菜单项交互完成，也可以通过 MATLAB 提供的各种命令完成。相比较而言，通过菜单交互的方法更简单便利，而通过 MATLAB 命令则可以达到更精细和强大的输出控制。

## 27.2　图形打印

### 27.2.1　使用菜单打印图形

MATLAB 中打印图形的菜单项主要包括 File 菜单下的 Page Setup…（页面设置）、Print Setup…（打印设置）、Print Preview…（打印预览）、Print（打印）四项。

一般情况下，用户顺次执行上述这四个菜单项，进行一系列自定义设置来完成打印作业。

选择页面设置菜单项可以打开页面设置对话框，在 MATLAB 命令窗口下运行 pagesetupdlg 命令也可以打开此对话框，如图 27-1 所示。页面设置对话框包括四个选项卡，它们分别为用户提供了对图形尺寸和位置、纸张、线条和文本、坐标轴和图形元素等选项的设置。

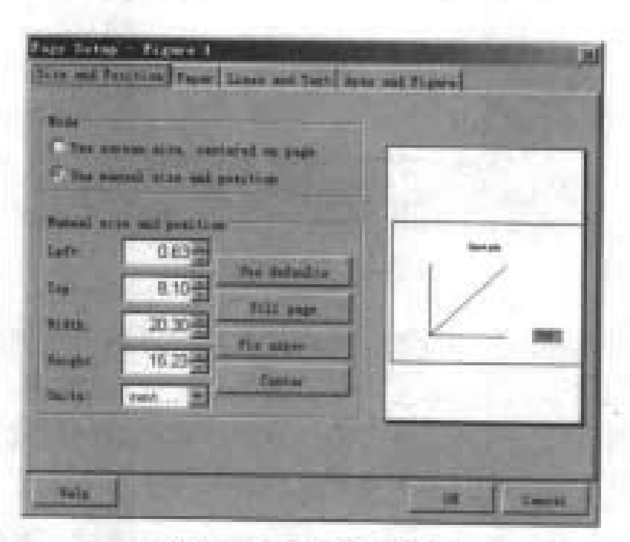


图 27-1　页面设置对话框

选择打印设置菜单项，将会弹出如图 27-2 所示的打印设置对话框。打印设置对话框中提供了打印机选择及打印机属性、打印纸张和打印方向的设置。



图 27-2　打印设置对话框

选择打印预览菜单项，或者运行 printpreview 命令，MATLAB 将会开启如图 27-3 所示的打印预览窗口。在此窗口下，用户可以看到实际打印效果的预览图，也可以通过此窗口提供的按钮进行页面设置、打印设置、页面头设置等，调整打印效果。

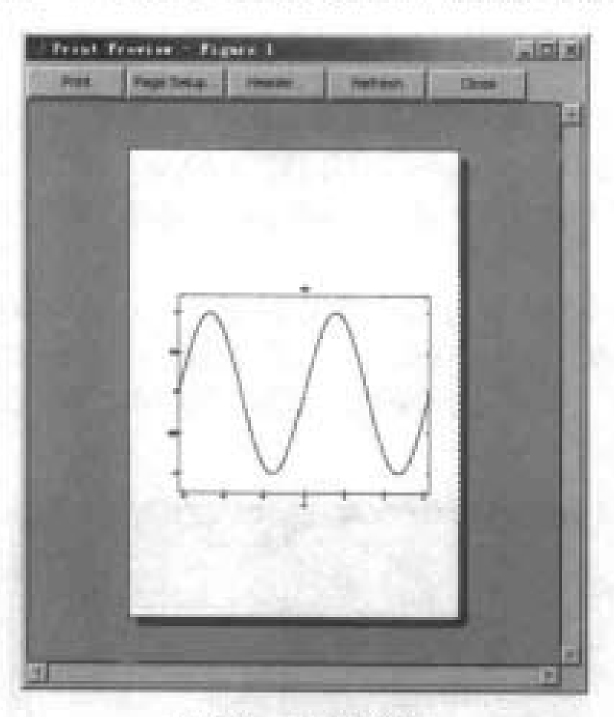


图 27-3　打印预览窗口

当用户完成了页面设置、打印设置，并通过打印预览窗口调整好了打印最终效果后，就可以选择 Print 项（或者执行 printdlg 命令）进行打印了。此时弹出如图 27-4 所示的打印对话框，用户需要完成打印机选择、打印范围和打印拷贝数的设置，然后单击确定按钮就可以按照用户自定义的设置完成打印作业了。



图 27-4　打印对话框

### 27.2.2　图形打印命令

MATLAB 中可以通过 print 命令将图形打印出来：





（1）print 直接打印 MATLAB 当前图形窗口；

（2）print filename 将图形打印到本地文件。

其他 print 的参数主要结合 set 命令实现复杂的打印作业。

set 命令设置的常用语法是：

set(gcf,PropertyName,'value')

print 命令可以接受多种设置参数。这两个命令综合使用，能实现细致的打印设置，如选择打印机、设置各种打印选项等，这些将在 27.2.3 中介绍。

### 27.2.3 打印设置

从 27.2.1 节中可以看到，完成特定的打印作业，用户需要进行许多打印相关的设置，包括设置图形尺寸和位置、图形中的线条和文本颜色转换、坐标轴范围和刻度标签、图形中的 UI 控制元素、图形着色器、打印纸张和方向、打印拷贝数、打印页面头等。这些都可以在页面设置对话框、打印设置对话框、打印预览窗口按钮或打印对话框中设置。当用户不自定义设置这些打印选项时，MATLAB 会根据默认设置项进行打印。

MATLAB 中图形打印和导出的默认设置是：

图形按照 8*6 英寸大小输出，纸张设置为 8.5*11 英寸大小（A4 页面）横向居中打印输出，取图形和坐标轴背景色为白色，自动缩放坐标轴范围和刻度标签适应打印页面大小。

下面介绍各打印选项的意义和设置方法。

#### 1．图形选择

通过菜单项交互打印时，MATLAB 打印当前窗口的图形。当有多个图形窗口开启时，一般是最后创建的窗口为当前图形窗口，当然用户可以通过鼠标点击切换当前图形窗口。

print 命令则可以通过 print –fhandle 这种调用选择图形窗口句柄为 fhandle 的图形窗口进行打印作业。

#### 2．打印机选择

通过图 27-2 的打印设置对话框或图 27-4 的打印对话框，用户可以选择要使用的打印机，选择其中的属性按钮可以进一步设置打印机属性。

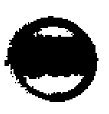
print 命令也可以指定打印机，其语法为：

print –pprinter

其中-p 参数说明了 printer 代表打印机名称，当打印机名称包含空格时，可以通过 print "–pyour print name"的方法来指定打印机。

#### 3．图形尺寸和位置

图形尺寸和位置，指 MATLAB 图形在打印纸张上显示的大小和位置，通过图 27-1 所示的页面设置对话框的图形尺寸和位置选项卡，可以进行设置。MATLAB 提供了两种图形尺寸和位置模式：



（1）第一种为屏幕模式，打印结果中图形尺寸和在显示器上显示的尺寸一致，并在页面居中；

（2）第二种为自定义模式，用户可以手动设置图形到页面左边缘和顶部的距离，以及图形自身的长度和宽度。

实际上在页面设置对话框下，用户可以拖动和调整窗口右边的预览缩略图中图形，完成对图形尺寸和位置的设置。

set 命令通过设置图形的 PaperPositionMode 属性，选择屏幕模式或自定义模式。在 PaperPositionMode 属性被设置为 manual，即选择了屏幕模式时，可以通过 set 设置 PaperUnits 和 PaperPosition 属性，指定表示图形位置和大小的单位和其具体取值，其常用语法为：

（1）set(gcf, 'PaperUnits', 'inches')

（2）set(gcf, 'PaperPosition', [left bottom width heigth])

#### 4. 页面尺寸和方向

通过页面设置对话框的页面选项卡（如图 27-5 所示），或者如图 27-2 的打印设置对话框，用户可以设置打印页面尺寸和打印方向。

MATLAB 中内置了多种具有特定尺寸的页面类型，一般情况下，用户都是通过选择页面类型来指定页面尺寸，如默认的 A4 页面类型，其宽 20.98 厘米，长为 29.68 厘米。打印方向表示了打印时页面和图形的相对方向，MATLAB 中提供了三种选择，一般用户都选择 Portrait（相当于横向）类型。

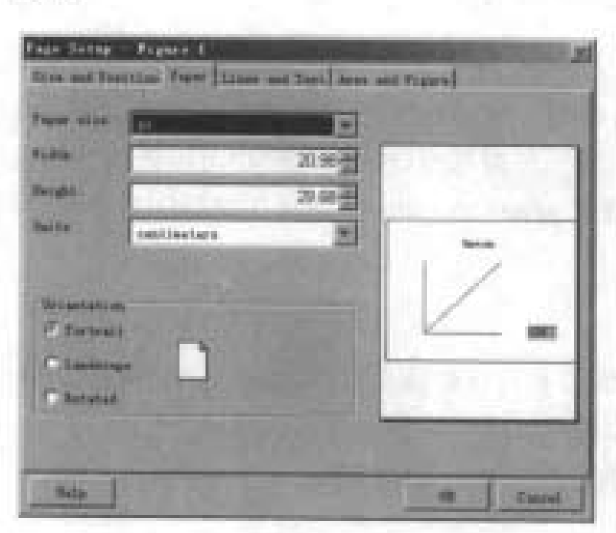


图 27-5 页面设置的页面选项卡

set 命令通过设置 PaperType 属性来指定页面类型，通过设置 PaperUnits 和 PaperSize 属性来自定义页面尺寸，通过设置 PaperOrientation 属性来指定页面方向。

#### 5. 线条和文本颜色

页面设置对话框的线条和文本选项卡（如图 27-6 所示）下，可以设置图形中的线条和文本的打印颜色。用户可以将图形中线条和文本的打印颜色转换为黑白色，或者不转换仍





按彩色打印。

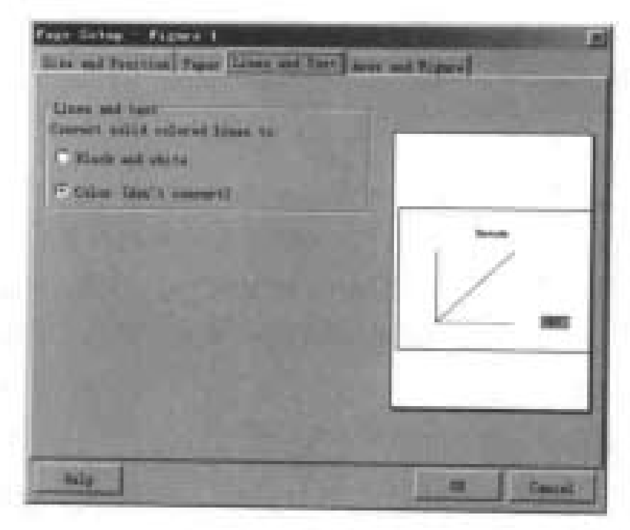


图 27-6　页面设置的线条和文本选项卡

需要注意，这些设置并不改变图形的显示效果，只是针对打印输出有效，但前提是打印机支持所选择的色彩模式。

用户也可以通过 set 命令来设置，如 set(findobj('type', 'line'), 'color', 'black')将线条颜色设置为黑白，set(findobj('type', 'text'), 'color', 'black')将文本颜色设置为黑白。

### 6. 坐标轴范围和刻度标签

一般情况下，用户设置的图形尺寸和屏幕显示的图形尺寸不会完全相同的，这就需要设置打印时对图形坐标轴范围和刻度标签的文字的处理方法。通过页面设置对话框的坐标轴和图形选项卡（如图 27-7 所示），用户可以指定打印时自动缩放调整坐标轴范围和刻度标签文字大小，来适应指定的图形大小，或者保持这些标签文字与屏幕显示的一致而不自动缩放。

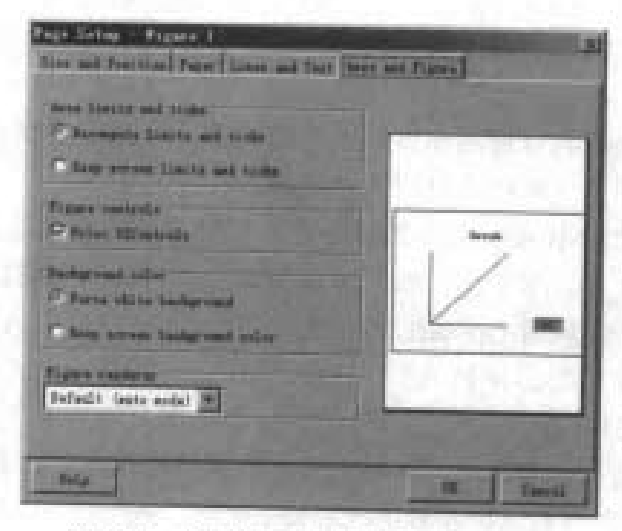


图 27-7　页面设置的坐标轴和图形选项卡



set 命令可以通过设置当前坐标轴对象的 XtickMode，YtickMode，ZTickMode 属性来改变某一坐标轴标签文字打印时的缩放模式，如 set(gca, 'XTickMode', 'manual')就设置 X 坐标轴标签文字自动缩放，适应图形尺寸。

### 7. 图形背景色

在图 27-7 所示的页面设置对话框的坐标轴和图形选项卡下，也可以设置图形打印时的背景色。用户可以选择强制使用白色背景或保持屏幕显示的背景色。

用户也可以通过 set 设置图形窗口的 InvertHardCopy 属性，指定是否保持屏幕显示的背景色。set(gcf, 'InvertHardCopy', 'off')禁止将图形背景色转变为白色，因此可以保持通过 set 设置的当前图形窗口的 color 属性。

### 8. 用户交互控制元素

用户交互控制元素，是指用户创建的，可以交互完成某些功能的图形界面交互控制元素。在图 27-7 所示的页面设置对话框的坐标轴和图形选项卡下，可以设置这些元素在打印时是否显示。默认情况下，这些元素会出现在最终打印效果中。

用户可以使用 print 命令的-noui 参数，指定不打印用户交互控制元素。

### 9. 选择着色器

着色器是将图形数据处理转换成适合显示、打印、导出格式的软件或硬件工具。MATLAB 中支持三种着色器：Painter's，Z-buffer 和 OpenGL。

（1）Painter's 使用向量绘图格式，能够产生高分辨率结果，当图形中只包含简单的图形对象时其处理速度最快，也是创建 PostScript 和 EPS 文件最好的着色器，但 Painter's 不能处理使用 RGB 色彩模式的片块或表面对象，也不能显示光照和透明效果。另外，对于采用 HPGL 打印驱动或要导入 Adobe Illustrator 文件的应用情况，Painter's 是惟一可选择的着色器。

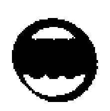
（2）Z-buffer 采用点阵绘图格式，运行速度一般比 Painter's 快，和 Painter's 相比能产生更精确的图像输出，能显示光照效果，但同样不能显示透明效果，并且在显示复杂图形时会消耗大量内存资源。

（3）OpenGL 也采用点阵绘图格式，一般情况下运行速度最快，能够显示光照和透明效果，可以配合某些图形处理硬件进行各种处理。

默认情况下，MATLAB 根据图形的复杂性、各种图形元素设置、打印驱动以及输出文件格式等自动选择着色器。对于点、线、区域和简单表面图，MATLAB 采用 Painter's 着色器；对于非真彩色显示，或者 OpenGL 被设为不可调用时，MATLAB 将会选择 Z-buffer 着色器；对于应用了光影效果的复杂图形，MATLAB 自动选择 OpenGL 着色器。

用户也可以在图 27-7 所示的页面设置的坐标轴和图形选项卡下，手动设置着色器。

对着色器的命令方式设置，既可以通过 set 设置当前图形窗口的 Renderer 属性完成，也可以通过指定 print 命令的运行参数完成。如 set(gcf, 'Renderer', 'zbuffer')或 print –zbuffer 都将当前图形窗口的着色器设为 Z-buffer。不过 set 函数设置将保存在当前图形中，而 print





指定参数只对该次打印作业有效。

### 10．分辨率

打印分辨率和打印效果是否精确清晰直接相关，不过需要注意点阵图和向量图的分辨率定义并不相同。MATLAB 中图形打印的默认分辨率和打印驱动程序相关，如表 27-1 所示。

表 27-1　MATLAB 默认打印分辨率

| 打印驱动 | 默认分辨率 |
|---|---|
| Windows 或 PostScript 驱动 | OpenGL 或 Z-buffer 模式时为 150dpi<br>Painter's 模式时为 864dpi |
| GhostScript 驱动 | OpenGL 或 Z-buffer 模式时为 150dpi<br>Painter's 模式时为 864dpi |
| HPGL 驱动 | 1116dpi（只能采用 Painter's 着色器） |

MATLAB 中用户也可以自行指定打印分辨率，一般在打印机属性设置里面都有相应设置项。而通过 print 命令的-r 参数也可以指定单次打印作业的分辨率，调用语法格式为 print –rnumber，如 print -r100 指定打印分辨率为 100dpi。

### 11．页面头

页面头是打印在图形顶部的说明性文字。设置页面头需要单击图 27-3 打印预览窗口中的 Header…按钮，此时弹出如图 27-8 的页面头设置对话框，其中用户可以设置页面头字符串、是否包含日期时间及其格式、字体等选项，在对话框底部还会给出初步的预览效果。

图 27-8　页面头设置

完成了这些设置后，用户只需要单击图 27-4 打印对话框中的确定按钮，就可以完成打印作业了。而命令方式的打印，则需要首先通过 set 完成各种设置，然后书写一条包含所有打印需用参数的 print 命令，完成一次打印作业。

实际上，print 命令还可以对打印作业进行更多设置，这里就不再赘述，有兴趣的读者可以参考 MATLAB 联机帮助文件。

# 27.3 图形导出

## 27.3.1 使用菜单导出图形

MATLAB 中图形导出的菜单项包括：File 菜单的 Export Setup…（导出设置）项、Edit 菜单的 Copy Options…（复制选项设置）项和 Copy Figure（复制图形）项。

通过导出菜单项，用户可以将 MATLAB 图形导出为特定标准格式的图像文件，在导出菜单项中，MATLAB 为用户提供了大量的导出设置选项。

通过复制选项设置和复制图形两个子菜单项，用户可以把 MATLAB 图形复制到 Windows 剪贴板，供其他应用程序直接粘贴使用。

## 27.3.2 图形导出命令

命令行模式下完成图形导出的命令还是 print，不过相应的语法格式有所改变：

（1）print –dfileformat 按照指定的图像格式把 MATLAB 图形复制到剪贴板；

（2）print -dfileformat filename 则把图形直接导出为指定格式、指定文件名的图像文件。

如 print -depsc myimage 就是把当前图形窗口的 MATLAB 图形导出为彩色的 myimage.eps 文件；而 print –depsc 则把当前图形以 eps 格式复制到 Windows 剪贴板供其他应用程序粘贴调用。

## 27.3.3 导出设置

### 1．导出到图像文件的设置

单击 File 菜单的 Export Setup…项，就会弹出如图 27-9 所示的导出设置对话框。这里提供了把 MATLAB 图形导出到文件的各种设置选项。



图 27-9 导出设置对话框



导出设置对话框包括了属性设置和导出样式设置，以及右边的一排操作按钮。

属性设置部分包括四部分：

（1）在 Size 属性下，用户可以设置由图形导出的图像文件的长宽尺寸；

（2）在 Rendering 属性下，用户可以设置图形导出时采用的色彩模式、着色器、分辨率和坐标轴标签文字、用户交互控制元素的显示模式；

（3）在 Fonts 属性下，用户可以设置图形中的文字导出时的字体、字号、倾斜角度等；

（4）在 Lines 属性下，用户可以设置图形中的线条导出采用的线型、线宽。

导出样式设置部分，用户可以把当前的属性设置保存为本地样式文件供以后重复调用，也可以调用之前保存的属性设置文件。

当用户调用了本地属性设置文件，或手动完成了各种属性设置后，就可以选择右边的 Apply to Figure 按钮将这些设置应用到图形窗口，然后选择 Export…按钮就可以把当前图形导出为图像文件了，此时弹出如图 27-10 所示的另存为对话框，选择合适的保存路径，指定图像文件名，并选择相应的保存类型后，就可以单击保存按钮，将当前 MATLAB 图形保存为特定格式的图像文件。



图 27-10　另存为对话框

MATLAB 支持多种标准的图像文件格式，图 27-11 中列出了另存为对话框中 MATLAB 支持的各种图像格式。

MATLAB Figure (*.fig)
Adobe Illustrator file (*.ai)
Bitmap file (*.bmp)
EPS file (*.eps)
Enhanced metafile (*.emf)
JPEG image (*.jpg)
MATLAB Figure (*.fig)
Paintbrush 24-bit file (*.pcx)
Portable Bitmap file (*.pbm)
Portable Document Format (*.pdf)
Portable Graymap file (*.pgm)
Portable Network Graphics file (*.png
Portable Pixmap file (*.ppm)
Portable inXmap file (*.pkm)
TIFF image (*.tif)
TIFF no compression image (*.tif)

图 27-11　MATLAB 支持的图像文件格式

通过 print 命令将图形导出为标准格式图像时的参数设置，和 27.2.3 中的大部分内容类

似，这里不再赘述，读者也可以参考 MATLAB 联机帮助。

2．复制到剪贴板的设置

选择 Edit 菜单的 Copy Figure 项将图形复制到 Windows 剪贴板之前，也经常需要进行相应设置。这可以通过 MATLAB 性能设置窗口下的图形复制模板和图形复制选项设置来完成。

选择 Edit 菜单的 Copy Options…项，可以打开如图 27-12 所示的图形复制选项设置窗口。




图 27-12　复制选项设置窗口

从图 27-12 可见，复制选项设置窗口可以设置图形复制到剪贴板时的格式、背景色和尺寸。MATLAB 把图形复制到剪贴板时只支持两种图像格式：

（1）彩色的增强图元文件向量图（Metafile）；

（2）8 位彩色 BMP 格式点阵图（Bitmap）。

MATLAB 默认自动选择复制模式，即能够使用 Metafile 时就使用 Metafile 格式。

MATLAB 性能设置的图形复制模板设置窗口中也有大量相关设置，如图 27-13 所示。

在图 27-13 所示的窗口中可以设置图形中文本文字、线条、用户交互控制元素复制时采用的格式。另外，MATLAB 专门为 Microsoft Word 和 Microsoft PowerPoint 应用程序中调用 MATLAB 图形提供了复制到剪贴板的专门设置样式。用户只需要单击图 27-13 顶部的 Word 按钮或 PowerPoint 按钮，然后单击底部的 Apply to Figure 按钮把这些专门的复制样式应用到图形窗口，再选择 Edit 菜单的 Copy Figure 项，就可以按照这些复制样式把 MATLAB 图形复制到剪贴板，这时在 Microsoft Word 和 Microsoft PowerPoint 应用程序就可以直接粘贴调用 MATLAB 图形了。



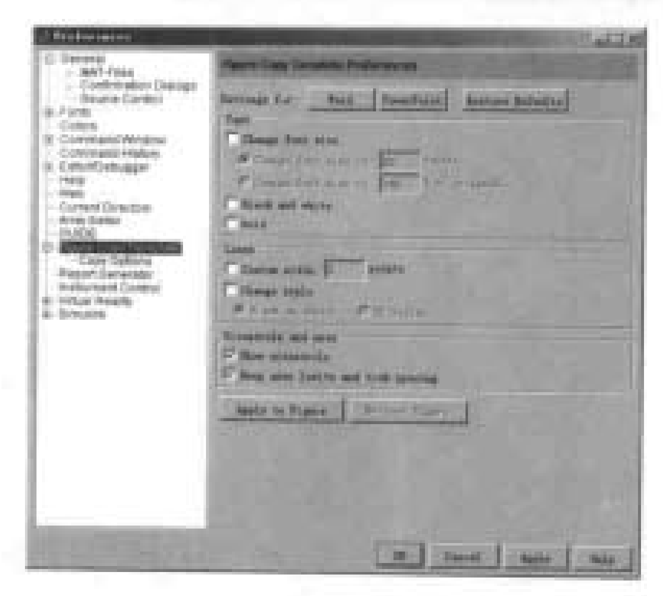


图 27-13　图形复制模板设置窗口

利用 print 命令也可以设置图形复制到剪贴板时的格式：

（1）print –dbitmap 指定 8 位彩色 BMP 格式点阵图复制格式；

（2）print –dmeta 指定彩色的增强图元文件向量图格式。

## 27.4　小结

本章介绍了 MATLAB 把图形打印输出和导出为标准格式的图像文件的方法，重点介绍了其中各种设置选项的意义和设置方法，一般用户对此部分内容能够了解和完成基本设置即可，对于有特殊应用的用户和 MATLAB 中高级编程用户，应该熟练掌握各种设置选项的图形界面设置方法和命令行设置方法。

# 第 28 章 句柄图形对象

MATLAB 是一种面向对象的高级计算机语言，其数据可视化技术中的各种图形元素，实际上都是抽象图形对象的实例，MATLAB 在创建这些图形对象实例时，会返回一个用于标识此对象实例的双精度浮点数值，称为该对象实例的句柄。通过操作句柄，用户就可以实现对象对应的图形对象实例的各种底层控制和设置。因此，这些对象也被称为句柄图形对象，本章讲述 MATLAB 中各种图形对象的组织形式，常用图形对象的基本属性和操作方法等。

## 28.1 句柄图形对象概述

前面的章节中介绍了可以在 MATLAB 的某个图形窗口的某个绘图子区的坐标轴中，创建二维、三维图线，三维曲面图，片块模型或者图像等。实际上，图形窗口、坐标轴、函数曲线、三维曲面、片块、图像等等，这些都是 MATLAB 数据可视化技术中用到的句柄图形对象。

句柄图形对象，是 MATLAB 为了描述某些具有类似特征的图形元素，而定义的具有某些共有属性的抽象的元素集合。如所有坐标轴元素都具有坐标轴范围、坐标轴刻度、坐标轴标签等属性，因此 MATLAB 中提供了坐标轴对象（Axes），它有许多属性项，创建一个坐标轴元素的过程，实际上就是将这些属性项赋值的过程。通过设置句柄图形对象的各种属性值，就可以实例化一个句柄图形对象，即创建了一个图形元素。句柄图形对象之于图形元素，很类似于各种面向对象的高级计算机语言中类和对象（类是对象的抽象集合，对象是对类的实例化）的关系。

MATLAB 中各种句柄图形对象是有层次的，其继承关系如图 28-1 所示。



图 28-1 中，垂直线条连接的两个对象具有父子继承关系，下层的对象继承自上层对象，或者说，下层对象是上层对象的子对象，上层对象是下层对象的父对象。一般地，子对象继承了父对象的所有属性，并且新添了许多独有属性。

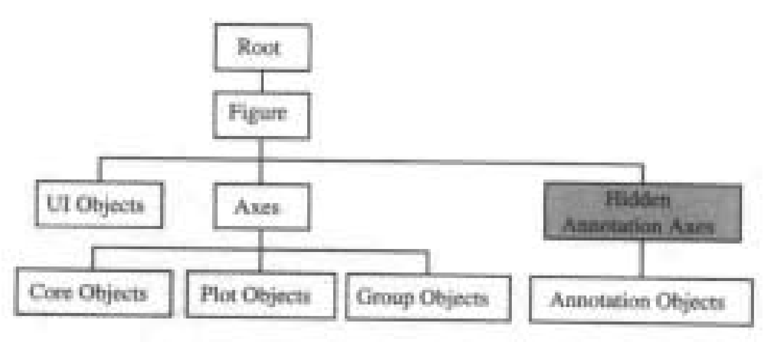



图 28-1 句柄图形对象继承关系

MATLAB 在创建各种句柄图形对象实例时，都会返回一个标识此对象实例的双精度浮点类型的数字标识，这称为该对象实例的句柄，通过图形对象句柄，用户可以实现对该对象实例的各种控制和设置。

MATLAB 绘图中，主要用到的是坐标轴对象（Axes）及其子对象，尤其是核心图形对象（Core Objects）。而 MATLAB 图形用户界面（GUI）设计中，则主要用到用户界面对象（UI Objects），这在本书 29 章会有介绍。本章后续章节将主要介绍根对象（Root）、图形窗口对象（Figure）和核心图形对象，以及对句柄图形对象的一般操作方法。

## 28.2 get 和 set 函数

各种句柄图形对象具有自己的属性，MATLAB 中可以使用 get 函数查询已创建句柄图形对象元素的各种属性，或者用 set 函数设置已创建句柄图形对象元素的各种属性：

（1）set(object_handle,'PropertyName','NewPropertyValue')将句柄为 object_handle 的图形对象元素的 PropertyName 属性取值设置为'NewPropertyValue'；

（2）returned_value = get(object_handle,'PropertyName')则查询句柄为 object_handle 的图形对象元素的 PropertyName 属性取值，返回给 returned_value。

## 28.3 根对象

在图 28-1 中，可以看到所有 MATLAB 的句柄图形对象都继承自根对象。

根对象只具有信息存储的功能，其句柄永远为 0。当启动一个 MATLAB 进程时，根对象随之产生，它负责存储 MATLAB 进程的状态、用户计算机系统的信息和 MATLAB 使用的默认值。表 28-1 列出了 MATLAB 根对象的常用属性。

表 28-1　根对象属性

| 属 性 名 | 取　值 | 意　义 |
| --- | --- | --- |
| CommandWindowSize | 二元数组 | MATLAB 命令窗口的宽、高取值 |
| CurrentFigure | 图形窗口句柄 | 当前图形窗口的句柄 |
| FixedWidthFontName | 字体名称 | 图形窗口下继承对象的字体 |
| Format | short，shorte（默认）等 | MATLAB 数值显示格式 |
| FormatSpacing | compact，loose（默认） | MATLAB 命令窗口显示的松紧模式 |
| Children | 句柄数组 | 根对象的所有子对象的句柄 |
| Parent | [] | 父对象句柄，根对象没有父对象，因此取值为[] |
| Diary | on，off（默认值） | MATLAB 操作记录开关 |
| DiaryFile | 字符串 | MATLAB 操作记录的文件名 |
| Echo | on，off（默认值） | MATLAB 命令行窗口的回显开关 |
| ErrorMessage | 字符串 | MATLAB 最后一格错误信息 |
| Tag | 字符串 | 对象标签，主要用于 find 查找对象句柄 |
| Type | root | 对象类型，根对象取值为 root（不可修改） |
| Units | pixels（默认），inches 等 | 度量单位 |
| Visible | on（默认），off | 对象可见开关，设置对根对象无效 |

因为根对象随 MATLAB 启动而产生，因此用户不能对根对象实例化，但用户可以通过 get 和 set 函数查询和设置根对象的某些属性。

例如 set(0,'Echo','on')就可以打开 MATLAB 命令窗口的回显模式，这样当运行 MATLAB 脚本 M 文件时，命令窗口会显示每一条命令及其输出结果。

## 28.4　图形窗口对象

图形窗口对象是根对象的直接子对象，所有其他句柄图形对象都直接或间接继承自图形窗口对象。

MATLAB 中可以通过 figure 函数实例化创建任意多个图形窗口对象，它们用于安置和显示各种句柄图形对象，因此，图形窗口对象实际上相当于容器。figure 函数的调用格式如下：

（1）不带参数的 figure 函数可以创建一个新的图形窗口，并将之设为当前图形窗口，MATLAB 一般返回一个整数（1，2，3 等）数值作为该图形窗口的句柄；

（2）figure(h)创建句柄为 $h$（必须是整数）的图形窗口，如果 $h$ 是一个已经存在的图形窗口的句柄，则将该图形窗口设为当前图形窗口；

（3）$h$ = figure(…)在创建图形窗口的同时返回该图形窗口句柄。

图形窗口对象继承自根对象，因此它具有所有根对象具有的属性，另外它也有许多独有的属性，其部分独有属性如表 28-2 所示。

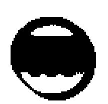


表 28-2 图形窗口对象属性

| 属性名 | 取　值 | 意　义 |
|---|---|---|
| Color | 颜色字符串或 RGB 三元数组 | 图形窗口的背景色 |
| Colormap | 颜色表数组 | 图形窗口的颜色表 |
| CurrentAxes | 坐标轴句柄 | 当前坐标轴句柄 |
| IntegerHandle | on（默认），off | 图形窗口对象句柄是否采用整数 |
| NextPlot | new，add（默认），replace，replacechildren | 下一图形对象添加模式 |
| Position | 四元数组 | 图形窗口在屏幕上所处的位置 |
| Renderer | painters，zbuffer，OpenGL | 图形窗口着色器 |
| RendererMode | auto（默认），manual | 图形窗口着色模式 |
| Resize | on（默认），off | 是否可以调整图形窗口大小 |
| WindowStyle | normal（默认），modal，docked | 图形窗口显示和捕捉模式 |
| CloseRequestFcn | 命令字符串或函数句柄 | 关闭图形窗口时运行的 MATLAB 命令 |
| CreateFcn | 命令字符串或函数句柄 | 创建图形窗口时运行的 MATLAB 命令 |
| ResizeFcn | 命令字符串或函数句柄 | 调整图形窗口大小时运行的 MATLAB 命令 |

MATLAB 中，用户可以通过 gcf 命令得到当前图形窗口的句柄，因此，gcf 也经常用在 get 或 set 函数中查询或设置当前图形窗口的属性，如 set(gcf,'Color','b')就将当前图形窗口的背景色设置为蓝色。

## 28.5 核心图形对象

MATLAB 中核心图形对象包括坐标轴、图线、表面、片块、光源、图像等，部分常用核心图形对象如表 28-3 所示。

表 28-3 核心图形对象

| 对象名称（创建函数） | 说　明 |
|---|---|
| axes | 坐标轴对象，定义图形显示区域的坐标系 |
| line | 图线对象，许多图线绘制函数都是创建图线对象 |
| surface | 三维表面对象 |
| patch | 片块对象，描述实体模型 |
| light | 光源对象，修饰其他图形对象的显示效果 |
| image | 图像对象，显示图像数组 |

坐标轴对象是许多图形对象的父对象，这从图 28-1 的继承关系图中可以看到。每一个可视化显示用户数据的图形窗口都包含一个或多个坐标轴对象，并且只有一个当前坐标轴对象，gca 函数返回当前坐标轴对象的句柄。坐标轴对象确定了图形窗口的坐标系统，所有绘图函数都会使用当前坐标轴对象或创建一个新的坐标轴对象，确定其绘图数据点在图形中的位置。

图线对象用于创建函数曲线，plot，plot3 等绘图指令都是对图线对象的实例的创建。



MATLAB 中通过各数据点的坐标及坐标系对象，确定数据点在坐标系中的位置，然后顺次连线创建出图线对象。

三维表面对象用于创建三维曲面，mesh，surf 及其派生函数实际上都是创建三维表面图像。其创建过程也是先确定数据点坐标对应的坐标轴中位置，然后连线和填充区域，从而创建三维表面对象。

片块对象是有边界的多边形填充区域，fill，fill3，patch 等函数都用于创建片块对象。

光源对象是不可见的图形对象，它用于修饰其他图形对象的显示效果。

图像对象是一个存储坐标系下每一个像素点颜色的数据数组，有时候也包括一个颜色表数组。本书第 26 章已经作过详细介绍。

表 28-3 中所列的各种核心图形对象都有各自独特的属性，读者可以参考 MATLAB 帮助文档查看详细介绍。

## 28.6 句柄图形对象操作

本书前面许多章节介绍了 MATLAB 中通过各种高级指令，如 plot，mesh，surf 等创建这些核心图形对象的方法。实际上，也可以用这些对象的低级创建函数（即对象名称，如表 28-3 所示），通过指定这些核心图形对象各自的属性来创建对象。

无论通过高级指令还是低级对象创建指令创建后的对象，都可以通过 get 函数获得其某属性的值，或者通过 set 函数设置某一属性取值。

在句柄图形的创建、属性查询、属性设置等操作中，对象句柄的应用是很广泛的。创建对象时经常需要指定其父对象句柄，用 set 函数设置对象属性时，经常要使用对象句柄标识被设置属性的具体对象，get 函数查询的某些属性返回结果也是句柄对象。因此，仔细理解和熟练应用这种通过应用句柄操作句柄图形对象的概念和方法，对于理解 MATLAB 中的句柄图形对象是必须的。下面就通过一个例子说明如何通过句柄操作句柄图形对象。

**例 28-1** 句柄图形对象操作。

**解：**在命令窗口输入：

```
%Ex28-1 create graphics by handles
clear
clc
close all
x=-2*pi:0.05*pi:2*pi;
y=sin(0.5*x).*cos(2*x);
fh=figure;
ah=axes('Parent',fh,'Layer','bottom','Position',[0.1 0.1 0.8 0.7], ...
    'XLim',[-8 10],'YLim',[-1 1],'XColor','b','YColor','b', ...
    'Box','on','XTick',[],'YTick',[]);
ahx=axes('Parent',fh,'Layer','top','Position',[0.1    0.1+0.5*0.7    0.8
eps], ...
    'XLim',[-8 10],'LineWidth',2.5,'XColor','m','XAxisLocation','top ' , ...
    'YColor','w','YTick',[]);
ahy=axes('Parent',fh,'Layer','top','Position',[0.1+0.8*8/18    0.1    eps
```





```
0.7], ...
        'YLim',[-1 1],'LineWidth',2.5,'YColor','r', ...
        'XColor','w','XTick',[]);
    line('XData',x,'YData',y,'Parent',ah,'Color','g')
    ahh=axes('Parent',fh,'Layer','top','Position',[0.68 0.12 0.2 0.2]);
    set(fh,'CurrentAxes',ahh);
    lh=ezplot(@(x)(sin(0.5*x).*cos(2*x)),[0.35 0.55]);
    set(get(ahh,'Title'),'String','zoomed x:[0.35 0.55]','Color','b');
    set(ahh,'XColor','b','YColor','b','Box','on','LineWidth',0.5, ...
        'XTick',[0.45],'XTickLabel','0.45','YTick',[0.13],'YTickLabel','
0.13', ...
        'XGrid','on','YGrid','on')
    set(get(ahh,'XLabel'),'String','')
    set(get(ahh,'YLabel'),'String','')
    set(lh,'LineStyle','--','Color','r')
    set(get(ah,'Title'),'String','Handle-Graphics','FontSize',25,'Color','b'
)
```

代码运行后，绘图结果如图 28-2 所示。

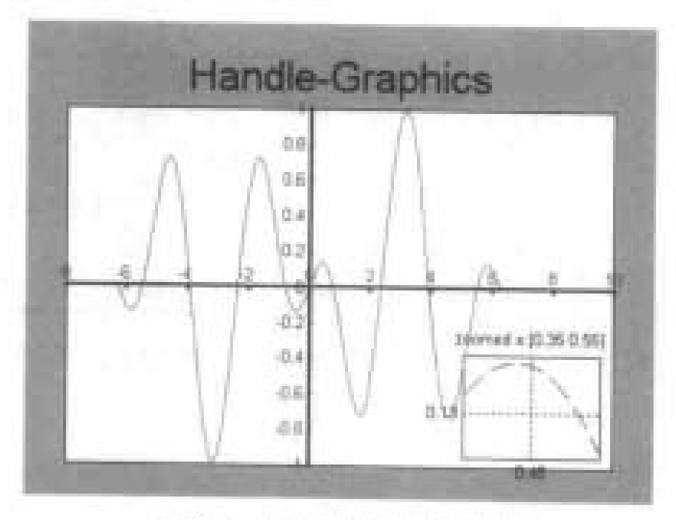



图 28-2　使用句柄图形对象绘图

例 28-1 中，通过低级对象创建指令和 set，get 函数，实现了如图 28-2 的绘图结果。这些基本的元素也都可以通过前面章节介绍的有关高级指令来创建，但通过低级指令，用户则可以更加精细地控制和设置这些元素。

关于 set 函数和 get 函数的调用方法本章 28.2 已经介绍，这里读者可以通过实际例子仔细体会它们的调用方法，尤其是通过句柄标识对象的方法，它是句柄图形对象操作中的核心。

## 28.7　回调函数

MATLAB 中，对句柄图形对象还可以设置一些事件响应函数，它们可以在对象创建、对象删除等事件发生时执行，从而实现特定事件触发下需要的某些功能，这些事件响应函

数，称为句柄图形对象的回调函数。

回调函数也是句柄图形对象的属性，其取值是字符串类型。当回调函数对应的事件发生时，MATLAB 通过 eval 函数执行回调函数的取值字符串，因此回调函数取值可以支持所有 MATLAB 合法命令语句、M 文件名或函数句柄。

所有的句柄图形对象都有表 28-4 所示的三个回调函数。

**表 28-4 所有句柄图形对象的回调函数**

| 回调函数 | 意　义 |
| --- | --- |
| ButtonDownFcn | 鼠标左键在对象上点击时执行的回调函数 |
| CreateFcn | 对象创建时执行的回调函数 |
| DeleteFcn | 对象删除时执行的回调函数 |

图形窗口对象支持更多的回调函数，如表 28-5 所示。

**表 28-5 图形窗口对象的回调函数**

| 回调函数 | 意　义 |
| --- | --- |
| CloseRequestFcn | 图形窗口接收到关闭请求时执行的回调函数 |
| KeyPressFcn | 光标处于图形窗口内，且用户按下某个按键时执行的回调函数 |
| ResizeFcn | 用户调整图形窗口大小时执行的回调函数 |
| WindowButtonDownFcn | 用户在图形窗口背景区、不可用控件或坐标轴背景区点击鼠标时执行的回调函数 |
| WindowButtonMotionFcn | 用户在图形窗口内拖动鼠标时执行的回调函数 |
| WindowButtonUpFcn | 用户在图形窗口内点击鼠标后再次释放鼠标按钮时执行的回调函数 |

默认情况下，表 28-5 所示的图形窗口对象的回调函数中，只有 CloseRequestFcn 是有默认设置值的，其他回调函数默认都为空字符串。

**例 28-2** 回调函数。

**解**：在命令窗口输入：

```
>> h=figure
h =
     1
>> get(gcf,'DeleteFcn')
ans =
     ''
>> str=get(gcf,'CloseRequestFcn')
str =
closereq
>> set(gcf,'CloseRequestFcn','')
>> close gcf
>> set(gcf,'CloseRequestFcn',str)
>> close gcf
```

例 28-2 中，各函数的功能介绍如下：

（1）h=figure 创建了一个图形窗口，返回其句柄为 1；

（2）get(gcf,'DeleteFcn')函数使用 get 命令查询句柄 gcf 标志的当前图形窗口对象的 DeleteFcn 回调函数，结果是空字符串，这表示默认情况下，删除图形窗口对象，MATLAB





并不执行额外的指令；

（3）str=get(gcf,'CloseRequestFcn')通过 get 命令查询当前图形窗口对象的 CloseRequestFcn 回调函数，结果返回给 str，输出结果显示其值是字符串 closereq，这表示当图形窗口接收到一个关闭请求时，MATLAB 会执行 eval('closereq')指令，即关闭接收到关闭请求的图形窗口；

（4）set(gcf,'CloseRequestFcn','')将当前图形窗口的 CloseRequestFcn 回调函数设为空字符串，这使得通过 close gcf 命令或者单击图形窗口的关闭按钮，都无法关闭图形窗口；

（5）重新把图形窗口对象的 CloseRequestFcn 回调函数设为字符串 closereq，才可以通过 close 命令或单击窗口关闭按钮来关闭图形窗口。

实际上，各种 MATLAB 的句柄图形对象都有其独有的回调函数属性，这些回调函数都以 Fcn 结尾，读者需要设置时，可以查看 MATLAB 中关于各种句柄图形对象属性的帮助内容，自行学习。

回调函数的一个更广泛的用途是设置图形用户界面控件的事件响应，这部分内容将在本书第 29 章中介绍。

## 28.8 小结

本章简单地介绍了 MATLAB 中数据可视化技术的底层概念——句柄图形对象，通过相应的对象创建函数和 get，set 函数，用户可以在最底层控制和设置句柄图形对象的各种属性。因此，可以说句柄图形对象操作，为用户操作图形元素提供了最精细的控制方法。

为了和本书前面章节配套，本章中只是较详细地介绍了与二维图形、三维图形等密切相关的根对象、图形窗口对象和核心图形对象。MATLAB 中操作句柄图形对象，最核心的是要理解对象句柄的概念和熟悉各种对象的属性及其意义。

关于句柄图形对象，另一个重要的概念是回调函数，这一属性为对象提供了事件响应机制，通过设置对象的回调函数，MATLAB 就可以在对象发生特定事件时，执行特定的代码段，实现一定的功能。这一方法主要应用在 GUI 界面设计中，本书第 29 章中将通过更加实用的例子讲述。



# 第 29 章 图形用户界面（GUI）

MATLAB 作为一种高级语言的集成开发环境，不但可以创建 M 文件那样的命令行方式运行的程序，也可以创建图形用户界面的程序。

图形用户界面的程序，是在图形界面下安排显示与用户交互的组件元素，用户可以通过鼠标点击、键盘操作这些交互组件，实现特定的功能，然后 MATLAB 的输出显示在程序界面中相应的结果显示区域中。因此，用户只和前台界面下的组件发生交互，而所有运算、绘图等内部操作，都封装在内部，终端用户不需要去追究这些复杂过程的代码。

图形用户界面编程，大大提高了终端用户使用 MATLAB 程序的易用性，因此，学习 MATLAB 图形用户界面编程，即 GUI 程序的创建，是 MATLAB 编程用户应该掌握的重要一环。

本章讲述 MATLAB 中通过 GUIDE 方式和命令行 M 文件方式创建 GUI 程序的方法，详细介绍 MATLAB 图形用户界面程序使用的各种交互组件，并对组件回调函数编程进行详细介绍。

## 29.1 GUI 和 GUIDE

### 29.1.1 GUI 程序概述

GUI，英文全名是 Graphical User Interface，直译为图形用户界面，它实际上是指这样的程序：用户可以在前台界面中通过一系列鼠标、键盘操作，指挥后台程序实现某些功能。这是一种大大提高程序易用性、交互性的计算机编程方法，很多高级语言，如 VC++、Java 都支持图形用户界面编程，MATLAB 也不例外。



在 MATLAB 中，GUI 编程和 M 文件编程相比，除了要编写程序功能的内核代码外，还需要编写前台界面。MATLAB 的图形用户界面程序的前台界面由一系列交互组件组成，这包括按钮、单选框、复选框、文本框、标签文字、滑动条等。MATLAB 把实现程序功能的内核代码和这些交互组件的鼠标或键盘事件关联起来，即通过设置这些交互组件的回调函数，来完成特定交互事件下后台程序完成的功能。

MATLAB 中，设计 GUI 程序的前台界面有全命令行的 M 文件编程和 GUIDE 辅助的图形界面设计两种方式。

（1）使用全命令行的 M 文件编程设计 GUI 程序界面，就是通过低级句柄图形对象创建函数，设置 GUI 界面下各个交互组件的属性。这主要用到句柄图形对象操作的方法，用户可以参考本书第 28 章和本章后续章节。

（2）使用 GUIDE 辅助设计是一种更简单的创建 GUI 程序界面的方法。GUIDE，英文全名是 Graphical User Interface development environment，即 MATLAB 提供的 GUI 程序的开发环境。GUIDE 环境，实际上就是一个图形用户界面程序，MATLAB 用户只需通过简单的鼠标拖拽等操作，就可以设计自己的 GUI 程序界面，因此也是一般用户实现 GUI 编程的首选方法。

## 29.1.2　打开 GUIDE 开发环境

GUIDE 开发环境，是 MATLAB 为 GUI 编程用户设计程序界面、编写程序功能内核，而提供的一个图形界面的集成化的设计和开发环境。

在 MATALB 主界面下，选择 File 菜单 New 子菜单下的 GUI 项，就会打开一个 GUIDE 快捷启动对话框，如图 29-1 所示，这一对话框下，用户可以选择创建一个新的 GUI 程序或打开已有的 GUI 程序。

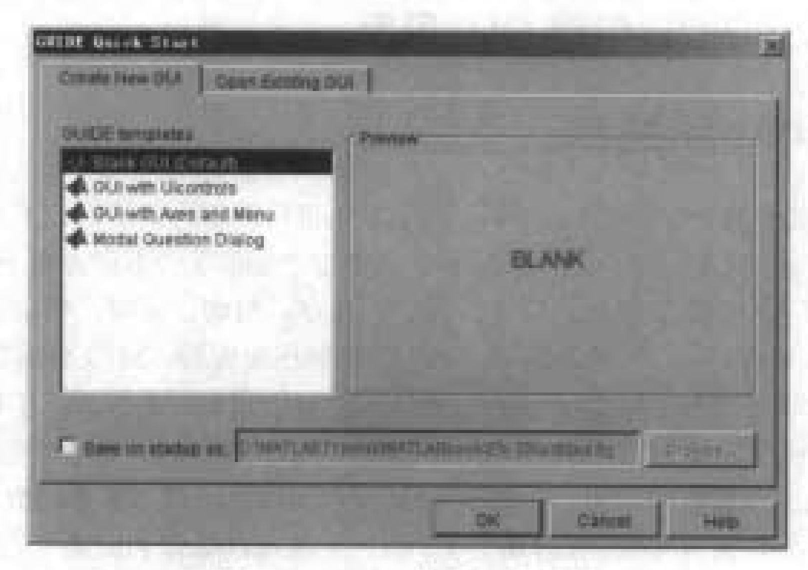



图 29-1　GUIDE 快速启动对话框

创建新的 GUI 程序时可以使用四种不同的 GUIDE 模板，如图 29-1 所示。

选择空白 GUI，将会打开如图 29-2 所示的空白的 GUIDE 设计界面。

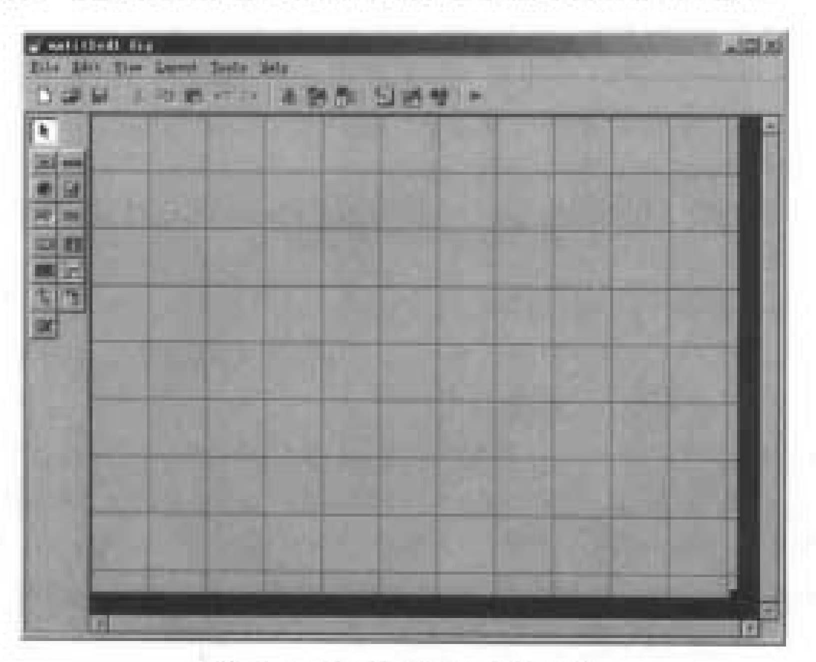


图 29-2　空白的 GUIDE 设计界面

在这一界面下，用户可以通过点击和拖拽鼠标的方式轻松创建自己的 GUI 程序界面。

在图 29-1 的新建 GUI 选项中，用户也可以选择其他模板项，这时候打开的 GUIDE 设计界面下就会有模板预先安排的一些交互组件，在有些情况下这可以减少用户设计界面的工作量，不过一般选择空白 GUI 模板就可以了。

下面介绍使用 GUIDE 环境创建 GUI 界面的方法。

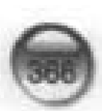
## 29.2　使用 GUIDE 创建 GUI 界面

### 29.2.1　GUIDE 界面概述

空白的 GUIDE 界面如图 29-2 所示。这一界面包括顶部的菜单栏、工具条，左侧边栏的组件面板和中心的 GUI 界面设计区域。其中菜单栏中提供了许多此界面下操作的菜单项。工具条中的按钮从左向右依次是：新建、打开、保存、剪切、复制、粘贴、撤销操作、返回撤销、对象分布和对齐、菜单编辑器、Tab 选择顺序编辑器、M 文件编辑器、对象属性设置窗口、对象浏览器和 GUI 运行按钮。其中，对象分布和对齐按钮，菜单编辑器按钮，M 文件编辑器按钮，对象浏览器按钮和 GUI 运行按钮在 GUI 设计中会经常使用。

左侧边栏的组件面板中，包括了 MATLAB 图形用户界面程序支持的所有交互组件，默认情况下按照小图标方式显示。实际编程中，用户可以通过选择 File 菜单的 Preferences 项，在弹出的性能设置对话框中选择 "Show names in component palette"，如图 29-3 所示。

选择 OK 后，GUIDE 界面下的交互组件面板将会显示各组件的名称，如图 29-4 所示。



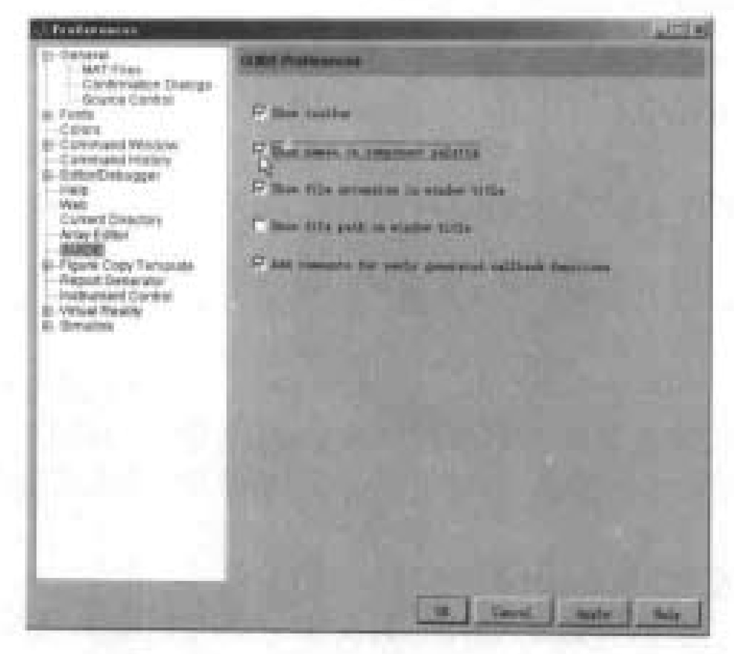


图 29-3　GUIUDE 界面性能设置对话框




图 29-4　显示组件名称的交互组件面板

GUIDE 界面的中间面积最大的区域，是 GUI 程序界面设计区，这一区域用来设计和显示用户 GUI 界面上的各种交互组件的位置。

## 29.2.2　交互组件

### 1．交互组件

如图 29-4 所示，MATLAB 中支持多种交互组件，它们包括：按钮（Push Button）、滑动条（Slider）、单选按钮（Radio Button）、复选框（Check Box）、文本框（Edit Text）、文本标签（Static Text）、下拉菜单（Pop-Up Menu）、下拉列表框（List Box）、海绵按钮（Toggle Button）、坐标轴（Axes）、面板（Panel）、按钮组（Button Group）、和 ActiveX 控件（ActiveX Control）。这些都是标准的图形界面程序常用的交互组件，它们适用于各种不同的应用场合。

（1）按钮是主要用于响应鼠标单击事件的交互组件，默认情况下，按钮处于上凸的弹起状态，当鼠标左键单击按钮时，按钮变为被按下的下凹状态，同时 MATLAB 响应按钮被单击的事件，当用户松开鼠标左键后，按钮又恢复为上凸的弹起状态。应用程序中常见的确定、取消等都用到按钮组件。

（2）滑动条主要用于为程序提供数值，这个数值被限制在一定范围内，用户可以通过鼠标或键盘移动滑动条上的方块的位置，来改变滑动条提供的数值。

（3）单选按钮经常是多个按钮一组联合使用，用于实现同一属性项在多项取值之间的切换。一组单选按钮在任何时间只能有一项被选定。单选按钮也用来为程序运行提供输入参数。

（4）复选框和单选按钮一样，也响应选定操作。和单选按钮不同的是，复选框提供互相独立的多项模式设置选项，一个复选框的选中状态不影响另一个复选框的状态。复选框

主要用于为程序运行提供模式选项。

（5）文本框支持用户通过键盘输入字符串，用于为程序运行提供输入参数。

（6）文本标签是显示固定字符串的标签区域，用于为其他组件提供功能解释和使用说明。

（7）下拉菜单类似于一组单选按钮，用户可以选择其中的一个项目来设置程序运行时需要的某个输入参数的取值。

（8）下拉列表框类似于一组复选框，用户可以选择其中的多个项目来设置程序运行需要的输入参数。

（9）海绵按钮类似于按钮，惟一不同的是用户单击一次海绵按钮后，其状态只能从上凸转换到下凹或者相反，而不是像按钮那样在释放鼠标后自动恢复到上凸状态。和按钮一样，海绵按钮主要用于响应鼠标单击事件，一般用于后台程序运行、终止或某些选项设置的生效等。

（10）坐标轴是图形化显示后台程序运行输出结果的区域。

（11）面板和按钮组是容器控件，用来把某些相关的交互控件组织在同一区域内，这样可以提高 GUI 界面的组织层次和易用性。

（12）ActiveX 控件主要用于 MATLAB 和其他应用程序的交互。本章不准备介绍这部分内容，有这方面需要的读者，可以参考 MATLAB 联机帮助了解学习。

#### 2. 添加交互组件

明确了各种交互组件的基本应用场合和功能，就可以在 GUIDE 界面下设计自己的图形用户界面了。这一过程主要是安排各种交互组件的位置并设置其属性，以及设计界面的菜单和快捷菜单。

用户首先应该根据 GUI 界面的基本设计确定设计区域大小，通过鼠标拖拽设计区域右下方的黑色方格位置就可以调整设计区域大小。

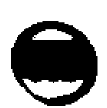
然后，就可以向 GUI 界面中添加交互组件了，这有两种方法可以实现：

（1）选择左侧栏面板中相应的交互组件，然后在设计区域中单击鼠标左键，就可以把该组件添加到设计区域的相应位置；

（2）用户也可以单击鼠标选择某种交互组件后，保持鼠标左键按下状态，然后拖拽移动鼠标到设计区域的相应位置后，松开鼠标左键，这样该组件就被添加到设计区域的相应位置。

添加到设计区域中的交互组件，用户可以通过鼠标操作改变其位置和大小。选择设计区域中的一个交互组件后，该组件四角会出现黑色的小方块，通过鼠标拖拽这些小方块就可以调整交互组件大小；在组件的其他位置单击并拖拽鼠标，可以改变组件在设计区域中的位置。

如图 29-5，创建了一个 GUI 界面，其中包括一个坐标轴（准备用于绘制图形），两个面板容器（分别放置模式选项和命令按钮），其中一个放置三个单选按钮（准备用于设置 shading 模式），另一个放置两个命令按钮（一个用于执行绘图命令，另一个清除坐标轴）。



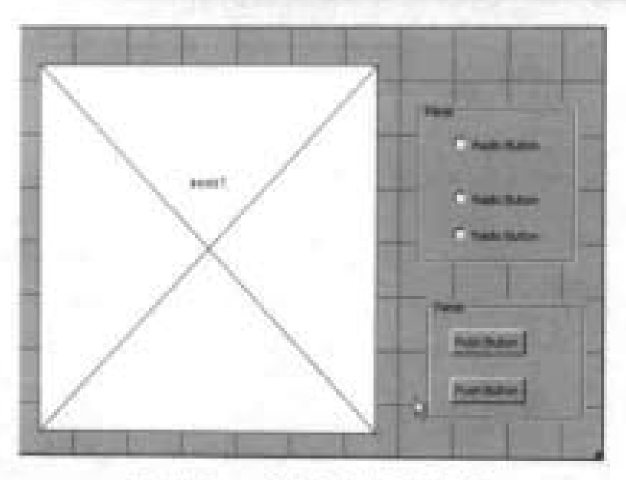


图 29-5 一个简单的GUI界面

### 3. 设置组件属性

创建了GUI界面需要的各个交互组件并调整大概的位置后，就需要设置这些组件的属性，双击组件会弹出该组件的属性设置窗口，如图29-6所示。

一般在属性设置窗口下完成各个组件的属性设置。



图 29-6 组件属性设置窗口

一般的GUI界面设计中，各种组件的常用属性包括颜色设置、文字标签设置、回调函数设置等。交互组件实际上也是句柄图形对象，因此其属性设置和第28章中讲解的各种句柄图形对象属性设置类似。

表29-1列出了各种交互组件（不包括坐标轴、面板和按钮组）通用的部分常用属性。

表 29-1 部分常用的交互组件通用属性

| 属　性 | 说　明 | 属　性 | 说　明 |
|---|---|---|---|
| BackgroundColor | 背景色（空间颜色） | Value | 组件当前取值 |
| ForegroundColor | 前景色（标签文字颜色） | SliderStep | 滑动条的步进增量 |

续表

| 属　性 | 说　明 | 属　性 | 说　明 |
| --- | --- | --- | --- |
| CData | 组件显示图像的数据数组 | Min | 最小值 |
| FontName | 字体名称（标签文字） | Max | 最大值 |
| FontSize | 字号 | ListboxTop | 下拉列表框顶部文字索引 |
| FontUnits | 字号单位 | Selected | 选中状态 |
| Position | 组件位置 | Visible | 可见状态 |
| String | 标签文字 | Enable | 可用状态 |
| Style | 交互组件种类 | Callback | 激活组件的回调函数 |
| Tag | 用户自定义的标记文字 | CreateFcn | 创建组件的回调函数 |
| Type | 'uicontrol' | ButtonDownFcn | 鼠标单击组件的回调函数 |
| Units | 单位 | KeyPressFcn | 组件上按键的回调函数 |
| HorizontalAlignment | 标签文字水平对齐方式 | DeleteFcn | 删除组件的回调函数 |

表 29-1 所列的属性中，Value，Min，Max 的取值依赖特定的组件种类，用户可以参考 MATLAB 帮助获得详细的解释。

大部分情况下，只需要设置组件的颜色属性、标签文字相关属性、Tag 属性和各种回调函数。Tag 属性主要用于查找获得对象句柄，回调函数是实现 GUI 程序内核功能的，本章最后将会介绍。因此，这里用户只需要设置好组件的颜色、标签文字等显示相关属性即可。

根据各组件的功能，按图 29-7 所示设置图 29-5 中创建的 GUI 界面下各个组件的属性。

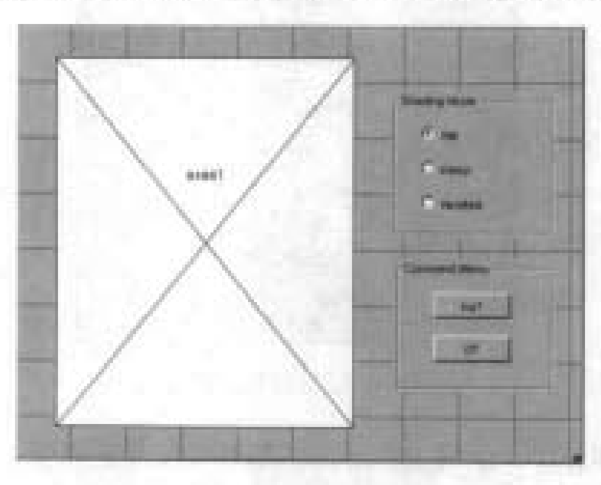
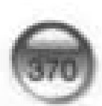


图 29-7　设置组件属性

#### 4. 组件的分布和对齐

创建需要的组件并设置它们的显示属性后，GUI 界面的创建就接近尾声了。最后还需要精细地调整组件的分布和对齐方式，使得最终产生的 GUI 界面整齐美观。

设置组件的分布对齐方式，首先要按下 ctrl 键，通过鼠标选择多个需要设置分布对齐方式的组件，然后单击工具条中的分布对齐设置按钮，弹出如图 29-8 所示的对象对齐设置窗口。



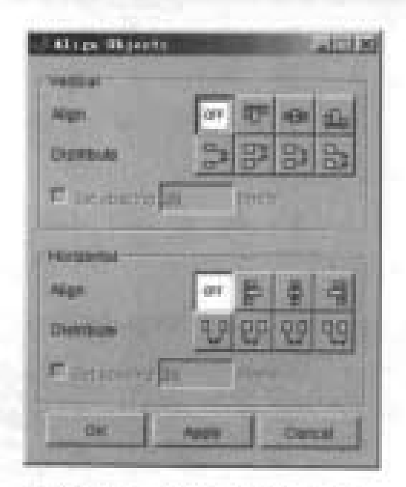


图 29-8　对象对齐设置窗口

图 29-8 的对象对齐设置窗口中，包括设置垂直对齐和分布、水平对齐和分布两个区域。各区域中按钮从左到右、从上到下其功能依次是：不设置垂直对齐方式、垂直方向顶端对齐、垂直方向中央对齐、垂直方向底端对齐；垂直方向组件间隔均匀、垂直方向组件顶端间隔均匀、垂直方向组件中央间隔均匀、垂直方向组件底端间隔均匀；不设置水平对齐方式、水平方向左对齐、水平方向中央对齐、水平方向右对齐；水平方向组件间隔均匀、水平方向组件左边缘间隔均匀、水平方向组件中央间隔均匀、水平方向组件右边缘间隔均匀。另外，在此对象对齐设置窗口下，也可以自定义指定各组件之间的水平、垂直方向的绝对间隔。

图 29-5 中创建的 GUI 界面设置完各组件的对齐和分布方式后，效果如图 29-7 所示。

### 29.2.3　设计菜单

一个标准的 GUI 界面，还应该包括普通菜单和右键弹出菜单。

在 GUIDE 界面下单击菜单编辑器按钮可以为 GUI 界面设计普通菜单和右键弹出菜单。菜单编辑器窗口如图 29-9 所示。



图 29-9　菜单编辑器窗口

从图 29-9 可以看到，菜单编辑器窗口底部有两个选项卡可以切换，分别用来设计 GUI 界面的普通菜单栏和右键弹出菜单。窗口顶部有一排按钮，从左向右依次是：新建菜单、新建菜单项、新建右键菜单、将选定的项在级别上向上移、将选定的项在级别上向下移、将选定的项在位置上向上移、将选定的项在位置上向下移、删除选定的项。窗口正中央显示当前创建的菜单和菜单项，窗口右侧则是当前选定项的属性设置区。

对图 29-7 所示的 GUI 界面，添加如图 29-10 所示的普通菜单。




图 29-10　创建菜单

## 29.2.4　GUI 程序的存储

当设计完 GUI 界面下的组件和菜单后，就应该保存这些设计工作了。选择保存工具按钮，出现另存为对话框，用户指定文件名后保存当前 GUI 设计即可。

GUIDE 默认情况下把用户的 GUI 设计保存成两个文件：

（1）一个是二进制的.fig 文件，用来存放搭建 GUI 界面所用的组件、菜单的属性；

（2）另一个是.m 文件，用来存放 GUI 程序响应特定事件时调用的函数。

保存按照图 29-7 设计组件和按照图 29-10 设计菜单后的 GUI 程序，指定文件名为 ex2901 后，则在当前目录产生两个文件：ex2901.fig 和 ex2901.m。

另外，用户也可以通过 File 菜单的 Export…项，把所有 GUI 程序设计代码存储在一个.m 文件中。这时弹出如图 29-11 所示的保存为单一.m 文件的对话框。







图 29-11 将 GUI 设计到处为单一.m 文件

比较 ex2901_export.m 文件和 ex2901.m 文件，可以发现前者除了包含后者所有内容外，在文件最后还多了一个函数 ex2901_export_LayoutFcn 用来创建 GUI 界面，这部分内容的开头部分如下：

```
% --- Creates and returns a handle to the GUI figure.
function h1 = ex2901_export_LayoutFcn(policy)
% policy - create a new figure or use a singleton. 'new' or 'reuse'.
```

接下来函数体的部分都是通过句柄图形对象的低级创建函数创建各个组件和菜单。对命令行 M 文件编程设计 GUI 程序界面感兴趣的读者，可以仔细学习这部分通过底层命令创建 GUI 界面的代码。

## 29.2.5 对象浏览器

搭建完 GUI 界面后，可以通过对象浏览器工具按钮，查看当前 GUI 界面下包含的所有组件、菜单项、快捷菜单项等，对象浏览器窗口还能显示这些对象的组织关系。

前文 GUI 例子对应的对象浏览器如图 29-12 所示。

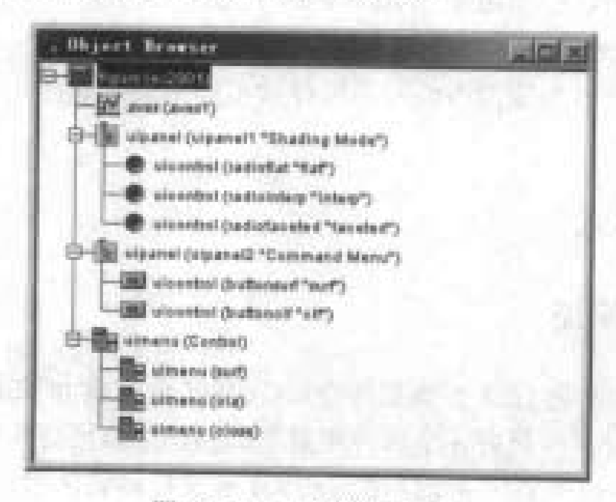
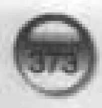



图 29-12 对象浏览器窗口

从图 29-12 可见，ex2901 这个 GUI 例子中，最顶层是一个图形窗口对象，包括一个坐标轴对象、两个面板对象和一个菜单对象，其中一个面板对象中容纳了三个单选按钮对象，



另一个面板中容纳了两个按钮对象，菜单对象下又包括了四个菜单项，这和前面创建过程是完全一致的。

### 29.2.6　GUI 程序的运行

搭建完 GUI 界面后，用户就可以通过工具条的 GUI 运行按钮来执行 GUI 程序。ex2901 这个 GUI 例子执行后出现如图 29-13 所示的 GUI 界面。

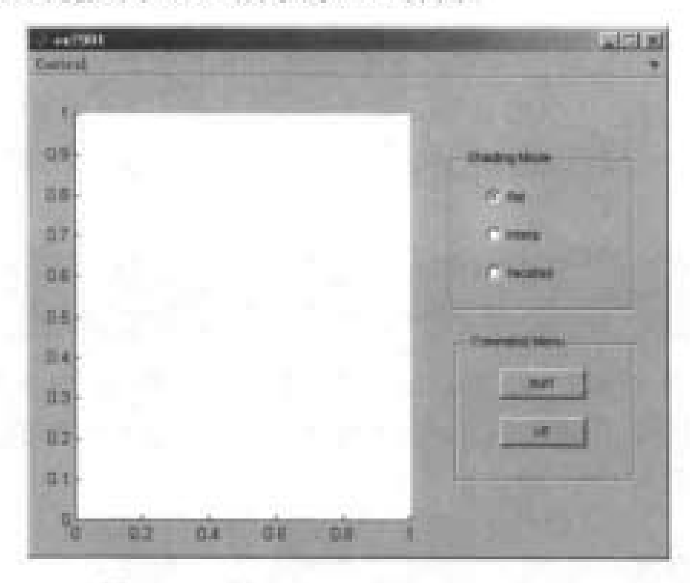


图 29-13　例子 ex2901 运行后的 GUI 界面

在图 29-13 的界面下，鼠标单击菜单、单选框、按钮，并没有任何预期的绘图效果出现，这是因为要使 GUI 程序响应用户的交互式操作，需要对各种组件和菜单编写事件响应的回调函数，而前面只是在 GUIDE 环境下完成了 GUI 界面的搭建，并没有编写各个组件的事件响应函数，因此这个程序是不能实现任何功能的。

下面就讲解在 M 文件编辑器中编写各组件和菜单的回调函数。

## 29.3　回调函数

### 29.3.1　回调函数原型

如前文所述，当搭建完 GUI 界面后保存，GUIDE 会生成.m 文件，用来存储程序的回调函数。但这时候.m 文件中只有各个组件和菜单的回调函数原型和注释，并没有实现功能的函数体。下面是只搭建了 GUI 界面后的 ex2901.m 文件除去注释内容的部分，可以看到，其中只有函数原型声明。

```
function varargout = ex2901(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```





```
                    'gui_Singleton',  gui_Singleton, ...
                    'gui_OpeningFcn', @ex2901_OpeningFcn, ...
                    'gui_OutputFcn',  @ex2901_OutputFcn, ...
                    'gui_LayoutFcn',  [] , ...
                    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function ex2901_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = ex2901_OutputFcn(hObject, eventdata, handles)

function radioflat_Callback(hObject, eventdata, handles)

function radiointerp_Callback(hObject, eventdata, handles)

function radiofaceted_Callback(hObject, eventdata, handles)

function buttonsurf_Callback(hObject, eventdata, handles)

function buttonclf_Callback(hObject, eventdata, handles)

function mnsurf_Callback(hObject, eventdata, handles)

function mncla_Callback(hObject, eventdata, handles)

function mnclose_Callback(hObject, eventdata, handles)

function mncontrol_Callback(hObject, eventdata, handles)
```

由此文件内容可见，回调函数名都是tagstr_Callback这种格式，其中tagstr是该对象的Tag属性的字符串值。输入参数中最重要的是hObject，代表该对象的句柄。

## 29.3.2 回调函数编程

要使GUI程序实现特定的功能，就需要为各交互组件、菜单等可以和用户交互的对象编写回调函数，使这些对象接收到用户的交互式操作时，完成特定的MATLAB命令。

下面以ex2901.m为例，逐一介绍每一个交互对象回调函数的编写。

在M文件编辑器中，用户可以单击函数显示按钮，这样可以选择文件中某个函数，如图29-14所示。选择某个函数后，当前光标会移动到相应位置的方便用户编辑该函数，这在文件中函数较多、文件很长的情况下是很有用的。

首先编写三个单选按钮的回调函数。这一回调函数主要实现单选按钮选中状态的排他性，即当三个单选按钮中的一个处于选中状态时，其他两个都转换到未选中状态。单选按

钮是否被选中状态，是由其'Value'属性值是否等于'Max'属性值决定的，当'Value'属性值等于'Max'属性值时，单选按钮处于选中状态，否则处于未选中状态。前面的GUI界面设计中设置属性时，已经通过这一方法将标签文字为flat的单选按钮设置为选中状态了。



图29-14　查看文件中的函数

单选按钮回调函数如下：

```
% --- Executes on button press in radioflat.
function radioflat_Callback(hObject, eventdata, handles)
% hObject    handle to radioflat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioflat
set(hObject,'Value',get(hObject,'Max'))
set(findobj(gcf,'Tag','radiointerp'),'Value', ...
    get(findobj(gcf,'Tag','radiointerp'),'Min'))
set(findobj(gcf,'Tag','radiofaceted'),'Value', ...
    get(findobj(gcf,'Tag','radiofaceted'),'Min'))


% --- Executes on button press in radiointerp.
function radiointerp_Callback(hObject, eventdata, handles)
% hObject    handle to radiointerp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiointerp
set(hObject,'Value',get(hObject,'Max'))
set(findobj(gcf,'Tag','radioflat'),'Value', ...
    get(findobj(gcf,'Tag','radioflat'),'Min'))
set(findobj(gcf,'Tag','radiofaceted'),'Value', ...
    get(findobj(gcf,'Tag','radiofaceted'),'Min'))

% --- Executes on button press in radiofaceted.
function radiofaceted_Callback(hObject, eventdata, handles)
% hObject    handle to radiofaceted (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiofaceted
set(hObject,'Value',get(hObject,'Max'))
```



```
set(findobj(gcf,'Tag','radioflat'),'Value', ...
    get(findobj(gcf,'Tag','radioflat'),'Min'))
set(findobj(gcf,'Tag','radiointerp'),'Value', ...
    get(findobj(gcf,'Tag','radiointerp'),'Min'))
```

函数的注释部分标注了各个函数执行的触发事件、函数参数意义以及相应对象的提示。在编写函数体时，主要用到句柄图形对象操作的 set 函数和 get 函数。findobj 函数则用来查找具有特定 Tag 值的对象句柄。

每一个单选按钮的回调函数的函数体中，都通过 set 设置了该按钮处于选中状态，同时把其他两个单选按钮设置为非选中状态。

接下来编写两个命令按钮的回调函数。其中 surf 按钮实现 surf 绘图功能，并且按照单选按钮选定状态确定图形 shading 模式；clf 按钮清除图形窗口下的所有对象。回调函数代码如下：

```
% --- Executes on button press in buttonsurf.
function buttonsurf_Callback(hObject, eventdata, handles)
% hObject    handle to buttonsurf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
hrf=findobj(gcf,'Tag','radioflat');
hri=findobj(gcf,'Tag','radiointerp');
hrc=findobj(gcf,'Tag','radiofaceted');
set(gcf,'CurrentAxes',findobj(gcf,'Type','Axes'))
ezsurf(@peaks)
if(get(hrf,'Value')==get(hrf,'Max'))
   shading flat
elseif(get(hri,'Value')==get(hri,'Max'))
   shading interp
elseif(get(hrn,'Value')==get(hrc,'Max'))
   shading faceted
end

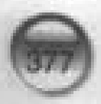
% --- Executes on button press in buttonclf.
function buttonclf_Callback(hObject, eventdata, handles)
% hObject    handle to buttonclf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clf
```

在 surf 按钮的回调函数体中，首先通过 findobj 函数获得三个单选按钮对象的句柄，然后通过 set 命令设置当前坐标轴，接着用 ezsurf 函数在当前坐标轴下绘图，最后通过 if 分支判断应该选择哪种 shading 模式。

在 clf 按钮的回调函数中，则仅执行 clf 命令，清除当前窗口中的所有对象。

最后编写三个菜单子项的回调函数。surf 菜单实现 surf 按钮一样的功能；cla 菜单清空坐标轴；close 关闭当前 GUI 窗口。函数体如下：

```
% --------------------------------------------------------------------
function mnsurf_Callback(hObject, eventdata, handles)
% hObject    handle to mnsurf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```



```
buttonsurf_Callback()

% --------------------------------------------------------------------
function mncla_Callback(hObject, eventdata, handles)
% hObject    handle to mncla (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla

% --------------------------------------------------------------------
function mnclose_Callback(hObject, eventdata, handles)
% hObject    handle to mnclose (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
```

surf 菜单的回调函数调用了 surf 按钮的回调函数（回顾本书第 10 章中的子函数概念）；cla 菜单执行 cla 命令清空坐标轴；close 菜单执行 close 命令关闭 GUI 窗口。

另外，control 菜单也有回调函数，但因为不需要它实现什么功能，因此函数体留空，如下所示：

```
% --------------------------------------------------------------------
function mncontrol_Callback(hObject, eventdata, handles)
% hObject    handle to mncontrol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

编写完所有的回调函数后，GUI 程序的编码阶段就完成了，这时候用户可以通过运行 GUI 程序，测试各项预期功能是否都已经实现，需要的话还可以进行代码调试和优化。

ex2901 的 GUI 示例程序预期实现如前所述的 shading 模式选择，surf 绘图和各种其他窗口、坐标轴控制。运行发现，经过以上的设计和函数编写后，此示例程序实现了预期功能。

## 29.4 小结

本章讲解了 MATLAB 中图形用户界面编程的技术，重点讲解了通过 GUIDE 环境搭建 GUI 程序界面和编写回调函数的方法。一般的 GUI 程序开发都遵循这样的过程。

本章所举例子虽然简单，但开发过程和其中应用到的概念和语法，则是典型的 GUI 程序开发共有的。希望用户能够追踪这一例子的创建过程，仔细体会本章中讲解的各种概念和方法。另外，建议读者参考本书第 10 章函数和第 28 章句柄图形对象，以便更好地理解本章内容。

# 第 30 章 MATLAB 类和面向对象编程

MATLAB 的面向对象编程使得用户可以定义新的数据类型，并对其进行一系列的操作而不需了解其完成的细节。

## 30.1 MATLAB 类概述

### 30.1.1 类的基本概念

MATLAB 中的类（class）与对象的概念，与 C++，Java 语言中的类和对象的概念类似，而且结构比其他语言要简单，更易于学习。类是一组具有相似特征对象（object）的抽象，是一种特殊的数据类型，表征对象的共性，类允许在其内部定义变量的数据结构和成员函数。

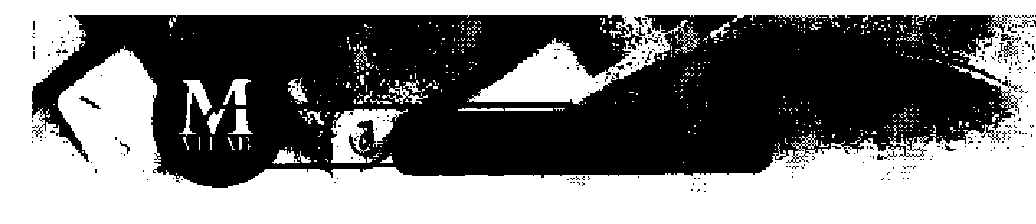
类对象是个体的，是类的一个实例。面向对象编程是指使用类和对象的编程方法。在 MATLAB 中可以将类简单地理解为，除基本数据类型之外的由 MATLAB 定义或用户定义的具有特殊功能的数据集合。

通过类，用户可以构造新型的 MATLAB 数据类型，并且声明一系列对该类类对象的运算符和操作函数。正是由于有了类，MATLAB 语言才真正具有了面向对象编程语言的特征，因此对类和对象概念的理解，是学习和使用 MATLAB 的关键。

所有的 MATLAB 数据类型都设计成面向对象编程中类的函数，图 30-1 表示了 MATLAB 种定义的 15 种基本的数据类型（或类），用户可以通过扩展类层次，添加新的数据类型到 MATLAB 中。

从图 30-1 中可以看出，用户类继承了结构类，因此用户创建的所有类都是基于结构的。

由于篇幅有限，关于类和面向对象的编程详细内容，请参考有关面向对象编程资料，此处只对重点内容作简要说明。



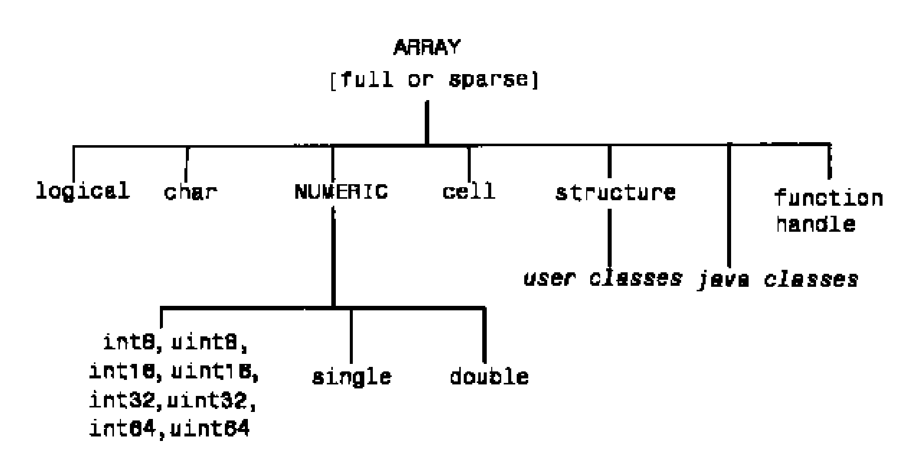



图 30-1　MATLAB 数据类型层次图

### 30.1.2　类的组成

通常一个类可以分为两个组成部分，即成员变量和成员函数：

（1）成员变量相当于 MATLAB 结构体的域，它可以为各种类型的 MATLAB 数据；

（2）成员函数是一系列的 M 文件，可以包括各种类型的 MATLAB 运算符、数学算子、功能函数，以及重载后的 MATLAB 内建算子和函数，用来对成员变量进行操作。

类对象的存储结构与前面讲到的 MATLAB 结构体的存储方式完全相同，但是在对域内容的访问和保护上却大不相同，由于数据的封装，类对象的域内容只能通过类的成员函数来获取和修改，外界无法访问。此外，在 MATLAB 系统下，类成员函数的存放有着严格规定，所有的类成员函数必须放在同一目录下，且目录名必须为@class_name(Windows 系统下)，其中符号@为固定格式，不可省略，class_name 为用户所定义类的类名；同时这个新建立的目录@class_name，必须设置为 MATLAB 系统搜索路径中工作目录的子目录。

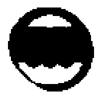
## 30.2　MATLAB 类的设计

### 30.2.1　在 MATLAB 中设计类的基本方法

在设计 MATLAB 的用户类时，一般使用标准方法，使类在 MATLAB 环境中协调地运行，当然也可以不使用标准方法。在不使用标准方法时，则要用定义大量的方法来完善类。表 30-1 列出了在 MATLAB 类中使用的基本方法。

表 30-1　MATLAB 类中使用的基本方法

| 类 方 法 | 描　述 |
| --- | --- |
| class constructor | 创建类的对象 |
| display | 显示对象的内容 |
| set, get | 访问类的特性 |
| subsref, subsasgn | 使自定义对象用下标进行引用或赋值 |





续表

| 类 方 法 | 描 述 |
| --- | --- |
| end | 在下标索引中支持关键字 end，如 A(1:end) |
| subsindex | 支持在表达式中使用对象 |
| 类似 char, double 的转换函数 | 将对象转换成 MATLAB 数据类型的方法 |

下面的部分将讲述类的这些方法的使用，并通过建立一个多项式类（polynom）的实例讲述 MATLAB 类的知识。多项式类表示带有一个行向量的多项式，该行向量包含按变量降幂排列的系数，因此，polynom 对象 $p$ 是只有包含系数的单一域（$p.c$）的结构，只有在@polynom 目录内的方法才能访问该域。

### 30.2.2 建立类目录

查看 MATLAB 版目录就会发现许多以@为前导的子目录，如 toolbox\mbc\mbcmodels\@localpoly，toolbox\mbc\mbcmodels\@pline 等。它们分别是 MATLAB 自己创建的符号函数、内联函数等“非内装”类的类目录（Class directory）。在这些目录下存放着定义类对象全部性质和操作方式的一组方法函数的 M 文件。

用户创建一个新类时，应该先为新类创建一个类目录。类目录的构成规则是：

（1）必须以@为前导符；

（2）@后面紧接待创建类的名称。

类目录的位置必须处在 MATLAB 搜索路径上某个目录的子目录的位置上。如果创建的类的目录不在 MATLAB 搜索的路径上，可用 addpath 命令将类的目录的父目录添加到路径中。本章中，将多项式类@polynom 建立在 testclass 子目录中，通过如下命令：

D:\MATLAB7\testclass\@polynom

可用下面的命令将类目录的父目录添加到 MATLAB 的路径中。

addpath D:\MATLAB7\testclass

如果创建的类的目录与其他类的目录同名，在调用类的方法时，MATLAB 认为这两个类的目录为同一个类的目录。

### 30.2.3 类的构造函数方法

类的构造函数必须执行，使得对象在 MATLAB 环境中正确运行的某些功能。通常，类构造函数必须处理三种可能的输入参数：

（1）无输入参数；

（2）与构造的类相同的类的对象作为输入参数；

（3）输入参数用来创建类的一个对象（通常是某种形式的数据）。

polynom 类构造函数@ polynom\ polynom.m 的代码如下：

```
function p = polynom(a)
%POLYNOM 多项式类构造函数
```

```
%   p = POLYNOM(v) 从向量 v 创建一个多项式对象，包含按 x 降幂排列的系数。
if nargin == 0
   p.c = [];
   p = class(p,'polynom');
elseif isa(a,'polynom')
   p = a;
else
   p.c = a(:).';
   p = class(p,'polynom');
end
```

构造函数调用语法如下：

1．无输入参数，如果构造函数不带输入参数，它返回一个不带字段的多项式对象；

2．输入参数为对象，如果构造函数的输入参数是一个多项式对象，MATLAB 返回输入参数，isa 函数检查这一情况；

3．输入参数为系数向量，如果输入系数是一个变量而不是一个多项式对象，将其改造成一个行向量，并分配给对象结构的.c 字段。类函数创建多项式对象，然后通过构造函数返回。

使用 polynom 构造函数的例子的语句为：

```
p = polynom([1 0 -2 -5])
```

以上语句创建一个带指定系数的多项式。

## 30.2.4　类的转换方法

转换方法将一个对象类转换成另一个类的对象，包含在 MATLAB 类中的两个最重要的转换方法是 double 和 char。double 转换方法产生 MATLAB 传统的矩阵，尽管这对于某些类可能是不恰当的。char 转换方法可以方便产生打印的输出。

### 1．polynom 到 double 的转换

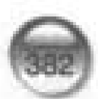
polynom 类的 double 转换方法是一个非常简单的 M 文件，@polynom/double.m，它仅仅取回系数向量。

polynom 类的 double 转换方法函数@ polynom/double.m 的代码如下：

```
function c = double(p)
% POLYNOM/DOUBLE  将 polynom 对象转换成系数向量。
%   c = DOUBLE(p) 将多项式对象转换成包含按 x 的降幂排列的系数向量 c。
c = p.c;
```

使用 polynom 类的 double 转换方法比较简单，在命令窗口中输入：

```
>> p = polynom([1 0 -2 -5])
double(p)
 %输出为
p =
    x^3 - 2*x - 5
 ans =
     1     0    -2    -5
```





### 2．polynom 到 char 的转换

polynom 类的 char 转换是一种重要的方法，因为它产生包含与变量 $x$ 独立的幂的字符串，因此，一旦指定 $x$，依照 MATLAB 正确表达的句法返回一个串，然后用户可以进行运算。

@polynom/char.m 的代码如下：

```
function s = char(p)
% POLYNOM/CHAR
% CHAR(p) 是串表示的p.c
if all(p.c == 0)
   s = '0';
else
    d = length(p.c) - 1;
    s = [];
    for a = p.c
        if a ~= 0;
            if ~isempty(s)
                if a > 0
                    s = [s ' + '];
                else
                    s = [s ' - '];
                    a = -a;
                end
            end
            if a ~= 1 | d == 0
                s = [s num2str(a)];
                if d > 0
                    s = [s '*'];
                end
            end
            if d >= 2
                s = [s 'x^' int2str(d)];
            elseif d == 1
                s = [s 'x'];
            end
        end
        d = d - 1;
    end
end
```

使用 polynom 类的 char 转换方法比较简单，在命令窗口中输入：

```
>> p = polynom([1 0 -2 -5]);  %创建polynom对象p
char(p) ;  %调用对象p的char方法
>> ans
%输出为
ans =
x^3 - 2*x - 5
```

char 返回的值是一个串，如果给出了 $x$ 的标量值，可以传递给 eval 函数，例如，接着输入：

```
>> x = 3;  %給x变量赋值
```

```
eval(char(p))  %计算多项式
%输出为
ans =
    16
```

### 30.2.5 类的显示方法

类显示方法通过 char 方法来产生多项式的一个串表示，然后在显示屏上显示该串。显示方法产生与标准 MATLAB 输出相同的输出。

@polynom/display.m 的代码如下：

```
function display(p)
% POLYNOM/DISPLAY 在命令窗口显示多项式。
disp(' ');
disp([inputname(1),' = '])
disp(' ');
disp(['   ' char(p)])
disp(' ');
```

例如在命令窗口输入：

```
>> p = polynom([1 0 -2 -5])  %创建一个 polynom 对象
%输出为
p =
   x^3 - 2*x - 5
```

### 30.2.6 类的 subsref 方法

假定设计 polynom 类指定一个对 polynom 对象的下标引用，

@polynom/ subsref.m 的代码如下：

```
function b = subsref(a,s)
% SUBSREF
switch s.type
case '()'
    ind = s.subs{:};
    for k = 1:length(ind)
        b(k) = eval(strrep(char(a),'x',num2str(ind(k))));
    end
otherwise
   error('Specify value for x as p(x)')
end
```

例如，在命令窗口输入：

```
>> p = polynom([1 0 -2 -5]);
p([3 4])  %分别对向量[3 4]的每个元素进行多项式运算。
%输出为
ans =
    16    51
```





## 30.2.7 类的重载

在许多情况下，当参数是对象时，需要改变MATLAB运算符和函数的功能，通常可以通过重载相应的函数完成。重载使得函数能处理不同类型和数目的输入参数，并以最高的优先级执行恰当的运算。

### 1. 重载多项式的算术运算符

每个内置的 MATLAB 运算符拥有相应的函数名（例如，加法运算符相应的函数名是plus.m）。用户可以通过在内目录中建立相应名称的M文件来重载任何操作符。

几个在多项式运算方面有意义的运算符需要应用到polynom类中。当重载算术运算符时，需要记住操作的数据类型。重载的运算符有‘+’，‘-’，‘*’，下面对其分别进行介绍。

（1）加法运算符重载

‘+’运算符重载的函数@polynom/plus.m的代码如下：

```
function r = plus(p,q)
% POLYNOM/PLUS  执行多项式的p + q。
p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] + [zeros(1,-k) q.c]);
```

函数首先确保两个输入参数都是多项式，然后访问两个系数向量，实际的加法仅仅是这两个向量的和，最后函数调用polynom构造函数来构造恰当的打印结果。

（2）减法运算符重载

‘-’运算符重载的函数@polynom/minus.m的代码如下：

```
function r = minus(p,q)
% POLYNOM/MINUS 执行多项式的 p-q。
p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] - [zeros(1,-k) q.c]);
```

（3）乘法运算符重载

‘*’运算符重载的函数@polynom/mtimes.m的代码如下：

```
function r = mtimes(p,q)
% POLYNOM/MTIMES   执行多项式的 p*q。
p = polynom(p);
q = polynom(q);
r = polynom(conv(p.c,q.c));
```

重载运算的例子如下：

```
>> p = polynom([1 0 -2 -5])
q = p+1   %重载加法运算符
r = p*q   %重载乘法运算符
%输出为
```

```
p =
    x^3 - 2*x - 5
q =
    x^3 - 2*x - 4
r =
    x^6 - 4*x^4 - 9*x^3 + 4*x^2 + 18*x + 20
```

表 30-2 列出了大部分 MATLAB 运算符的重载函数名称。

表 30-2 常用的 MATLAB 运算符重载函数

| 运 算 符 | M 文件名 | 描 述 |
|---|---|---|
| *a*+*b* | plus(*a*,*b*) | 二元加法 |
| *a*-*b* | minus(*a*,*b*) | 二元减法 |
| -*a* | uminus(*a*) | 一元减法 |
| +*a* | uplus(*a*) | 一元加法 |
| ***a*.*b*** | times(***a*,*b***) | 点乘法 |
| ***a**b*** | mtimes(***a*,*b***) | 矩阵乘法 |
| ***a*./*b*** | rdivide(***a*,*b***) | 点右除 |
| ***a*.*b*** | ldivide(***a*,*b***) | 点左除 |
| ***a*/*b*** | mrdivide(***a*,*b***) | 矩阵右除 |
| ***a**b*** | mldivide(***a*,*b***) | 矩阵左除 |
| ***a*.^*b*** | power(***a*,*b***) | 点指数 |
| ***a*^*b*** | mpower(***a*,*b***) | 矩阵幂 |
| *a* < *b* | lt(*a*,*b*) | 小于 |
| *a* > *b* | gt(*a*,*b*) | 大于 |
| *a* <= *b* | le(*a*,*b*) | 不大于 |
| *a* >= *b* | ge(*a*,*b*) | 不小于 |
| *a* ~= *b* | ne(*a*,*b*) | 不等于 |
| *a* == *b* | eq(*a*,*b*) | 等于 |
| *a* & *b* | and(*a*,*b*) | 逻辑与 |
| *a* | *b* | or(*a*,*b*) | 逻辑或 |
| ~*a* | not(*a*) | 逻辑非 |
| *a*:*d*:*b*<br>*a*:*b* | colon(*a*,*d*,*b*)<br>colon(*a*,*b*) | 冒号运算符 |
| ***a***' | ctranspose(***a***) | 复数共轭转置 |
| ***a***.' | transpose(***a***) | 矩阵转置 |
| command window output | display(***a***) | 显示方法 |
| [*a* *b*] | horzcat(***a*,*b***,…) | 串联运算 |
| [*a*; *b*] | vertcat(***a*,*b***,…) | 并联运算 |
| *a*(*s*1,*s*2,…*sn*) | subsref(*a*,*s*) | 下标引用 |
| ***a***(*s*1,…,*sn*) = b | subsasgn(*a*,*s*,*b*) | 下标指定 |
| ***b***(*a*) | subsindex(*a*) | 下标索引 |



2. 重载多项式类的函数

MATLAB 已经具有几个对以系数向量表示的多项式运算的函数，它们也需要重载以用于新的多项式类中。

（1）重载多项式类的求根

多项式对象求根的方法函数@polynom/roots.m 的代码如下：

```
function r = roots(p)
% POLYNOM/ROOTS.  ROOTS(p)是包含 p 的根的向量。
r = roots(p.c);
```

重载的多项式求根例子如下：

```
>> p = polynom([1 0 -2 -5]);
roots(p)
%输出为
ans =
   2.0946
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i
```

（2）重载多项式类的求值

多项式求值的方法函数@polynom/ polyval.m 的代码如下：

```
function y = polyval(p,x)
% POLYNOM/POLYVAL  POLYVAL(p,x) 计算多项式 p 在点 x 处的值。
y = 0;
for a = p.c
   y = y.*x + a;
end
```

（3）重载多项式类的画图

多项式画图的方法函数@polynom/ plot.m 的代码如下：

```
function plot(p)
% POLYNOM/PLOT  PLOT(p) 绘制多项式 p。
r = max(abs(roots(p)));
x = (-1.1:0.01:1.1)*r;
y = polyval(p,x);
plot(x,y);
title(char(p))
grid on
```

（4）重载多项式类的微分

多项式微分的方法函数@polynom/ diff.m 的代码如下：

```
function q = diff(p)
% POLYNOM/DIFF  DIFF(p) 是多项式 p 的微分。
c = p.c;
d = length(c) - 1;  % degree
q = polynom(p.c(1:d).*(d:-1:1));
```

### 30.2.8 类方法综合使用实例

对于某一特定的类，可以通过下列方式显示该类的所有方法：

（1）调用函数 methods('类名')；

（2）在命令窗口输入 methods 类名。

例如，在命令窗口输入：

```
>> methods('polynom')
%输出为
Methods for class polynom:
char      display  minus    plot      polynom  roots
diff      double   mtimes   plus      polyval   subsref
```

**例 30-1** 创建两个多项式 $x$ 和 $p=x^3-2x-5$，并计算 $p^2+10p+20x$ 的微分，最后绘制该微分曲线减去 20 的曲线。

**解**：在命令窗口中输入：

```
>> x = polynom([1 0]);   %构建多项式类 x
p = polynom([1 0 -2 -5]); %构建多项式类 p
plot(diff(p*p + 10*p + 20*x) - 20)  %使用重载运算符及函数进行运算
```

输入上述命令后，输出图 30-2。

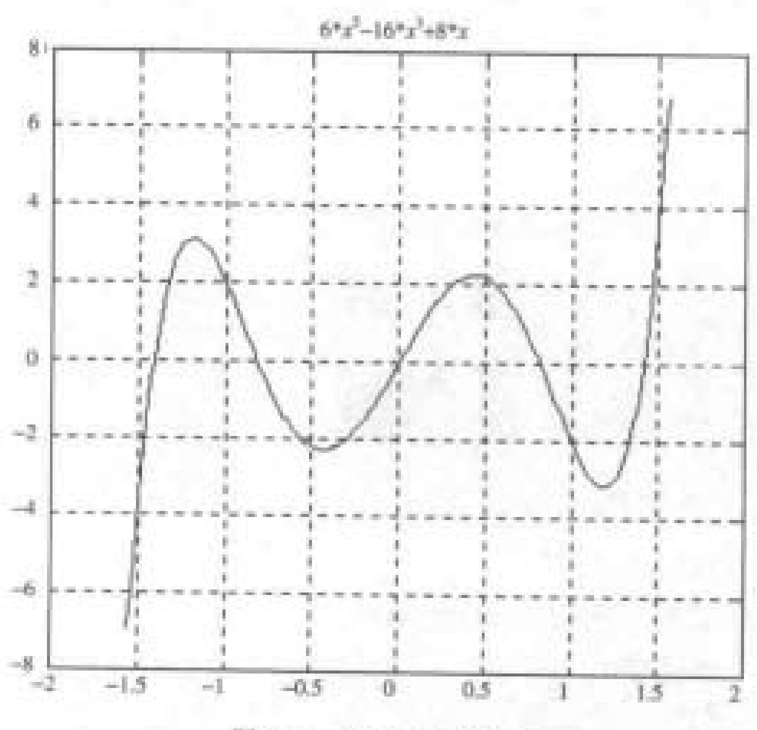



图 30-2 例 30-1 的输出图

```
%接着输入
>> x
%输出为
```





```
x =
   x
%接着输入
>> p
%输出为
p =
    x^3 - 2*x - 5
%接着输入
>> p*p + 10*p + 20*x
 %输出为
ans =
    x^6 - 4*x^4 + 4*x^2 + 20*x - 25
 %接着输入
>> diff(p*p + 10*p + 20*x)
%输出为
ans =
   6*x^5 - 16*x^3 + 8*x + 20
```

## 30.3 MATLAB面向对象编程

### 30.3.1 MATLAB面向对象编程的特点

运用类和面向对象编程，可以提高代码的重用性，避免了编程人员的重复劳动，提高了编程效率，并使程序易于维护和扩展。与常规的结构化编程相比，MATLAB的面向对象编程具有以下几个特点：

（1）函数与运算符的重载

用户可以创建覆盖现有MATLAB函数的方法。当调用一个以用户定义的对象为参数的函数时，MATLAB首先检查是否有对此对象的类定义的方法，如果存在的话，MATLAB就调用该方法，而不去调用通常的MATLAB函数。这种用类的方法优先于标准函数执行的方法，称为函数的重载。对于MATLAB运算符，同样也可以重载，即在类中定义同一运算符的各种不同功能。

（2）数据和方法的封装性

封装就是将数据和操作数据的函数衔接在一起，构成一个具有类类型的对象。对象的特性在MATLAB命令行中是不可见的，只能通过类方法来访问对象的特性。这样能够保证类中的数据不被其他非专用对象的类访问，从而提高了数据操作的安全性。

（3）继承性

用户可以建立类与类之间的层次关系，如子类与父类关系。子类从父类中继承数据域和方法，它可以继承单个父类的内容，也可以继承多个父类的内容。继承关系可以是多层次的。继承的子类可以共享父类的函数，以便子类中相同函数的功能统一。MATLAB的用户类都是结构体数组类的子类。数组类是MATLAB中基本的数据类型，它是虚拟类，不能用它创建对象。



（4）聚合

用户可以使用聚合来创建类，其中对象包含其他的对象。当对象类型是另一个对象类型的一部分时，这是合适的，例如，一个储蓄账户对象可能是一个金融组合对象的一部分。

### 30.3.2 MATLAB 面向对象编程与其他语言对比的特点

与 C++和 Java 等面向对象语言相比，MATLAB 的面向对象的特点的重要区别在以下几个方面。

（1）在 MATLAB 中，方法的指定不是基于特定的语法格式。在 C++和 Java 中，参数列表中包括的对象的优先级相等。MATLAB 中使用最左边的对象调用相应的方法。

（2）在 C++和 Java 中使用析构函数释放对象占用的内存，在 MATLAB 中没有析构函数的概念，从工作空间中删除一个对象仍然使用命令 clear。

（3）MATLAB 的数据类型在运行时被构造，而不是在编译时被构造。通过调用 class 函数，注册属于这个类的对象。

（4）使用 MATLAB 的继承关系创建类时，继承关系定义在子类中，通过创建父类的对象建立继承关系。子类的对象中保留一个与父类对象特性相同的类，并且在子类中的父类对象与父类的名称相同。

（5）MATLAB 不通过引用进行变量传递。当编写完一个更新对象的方法后，用户必须传递回更新的对象，并使用赋值表达式。例如使用 set 方法更新一个对象 A 的 name 域，并返回更新后的对象，则要用以下表达式：

```
A = set(A,'name','John Smith');
```

MATLAB 在处理它的类与对象时，采用概念上的面向对象的方法，在具体的处理方式上与完全意义上的面向对象程序设计语言相差很大。如果用户对其他面向对象的语言很了解，还应注意以下几个方面：

（1）在 MATLAB 中，没有与抽象的类等价的概念；

（2）在 MATLAB 中，没有等价于 Java 序的接口；

（3）在 MATLAB 中，没有等价于 C++的范围操作符；

（4）在 MATLAB 中，没有虚拟的继承和虚拟的基类；

（5）在 MATLAB 中，没有类似于 C++模板的概念。

## 30.4 小结

本章讲述了 MATLAB 类的基本概念，重点讲述了 MATLAB 中类的设计，以及面向对象编程，为读者掌握在 MATLAB 环境中创建类以及面向对象编程打下基础。



# 第 31 章 MATLAB 编程接口

MATLAB 具有强大的计算能力、可视化功能、便捷的开放式可扩展环境和面向各研究领域专用的工具箱（即数学函数包），这些数学函数包，不仅大大方便了特定领域的用户，而且使得 MATLAB 易学易用。同时，MATLAB 提供了强大的编程接口，支持 MATLAB 与其他应用程序进行数据交换，使其他应用程序可以利用 MATLAB 的强大功能。

## 31.1 MATLAB 与 Excel 接口

Excel 是 1985 年由美国微软公司推出的电子表格处理软件，它内置较多的函数集用于数据处理，并以图、表相结合的形式广泛应用于统计和分析等科学计算领域。

MathWork 公司开发的 MATLAB 软件具有强大的计算功能，虽然 MATLAB 和 Excel 功能有所交叉，但 MATLAB 强大的矩阵运算能力、灵活的可编程能力、完美的三维图形输出能力及丰富的工具箱函数，使得数据在 MATLAB 中处理比在 VC 或 VB 中处理更为简洁和节省编程时间，但 Excel 所具有的独特的图表功能（如散点图等），和其特有的日期和时间等函数的结合使得数据的二维图形显示更为方便和易于分析。因此，充分利用这两种软件的优点对数据进行处理将更加方便。

MATLAB 与 Excel 有两种接口方式：

（1）通过 MATLAB 提供的 Excel 生成器生成 DLL 组件和 VBA 代码，实现 Excel 对 MATLAB 的调用；

（2）通过 MATLAB 提供的 Excel link 插件，直接在 Excel 环境下运行 MATLAB 命令，与 MATLAB 进行数据传输。

下面对应用较多的第二种方式进行介绍。

### 31.1.1 Excel link 的使用

MathWork 公司开发的 MATLAB Excel link 成功地把 Microsoft Excel 和 MATLAB 集成在一起，为表格处理、科学计算和工程设计营造了一个完美统一的工作环境。它不仅具备 Excel 的全部功能，而且还具备 MATLAB 无与伦比的数学运算能力和灵活自如的数据可视化能力。微软公司的 Excel 在表格处理上占绝对优势，而 MATAB 在数值计算中占绝对优势，若将两者结合，其功能的强劲更是无与伦比。

MATLAB 中 Excel link 链接技术实现 Excel 与 MATLAB 的动态链接，用户可以在 Excel 空间里，利用 Excel 的宏编程工具，使用 MATLAB 的数据处理与图形处理功能进行操作。在使用时，不必脱离 Excel 环境，而是直接在 Excel 工作区或宏操作中调用 MATLAB 函数。

Excel link 是一个在 Windows 环境下实现 Excel 与 MATLAB 连接的插件，使用 Excel link 时，可以不必脱离 Excel 环境，直接在工作表中或宏操作中调用 MATLAB 函数，Excel link 同时保证两个工作环境中数据的交换和同步更新。

#### 1. Excel link 的安装

Excel link 安装步骤如下：

（1）启动 Excel；

（2）单击工具加载宏命令，在弹出的加载宏对话窗口中单击浏览按钮；

（3）在 D:\MATLAB7\toolbox\exlink（假设 MATLAB 安装在 D:\下）下选中文件 exlink.xla；

（4）返回加载宏对话窗口，单击确定按钮，Excel link 则被加载到 Excel 中。

这时 Excel 的工具栏上就会出现 startmatlab，putmatrix，getmatrix，evalstring 四个按钮，如图 31-1 所示。

这四个按钮的功能如下。

startmatlab：用于在 Excel 中启动 MATLAB；

putmatrix 和 getmatrix：用于 Excel 与 MATLAB 交换数据；

evalstring：用于在 Excel 中执行 MATLAB 命令。

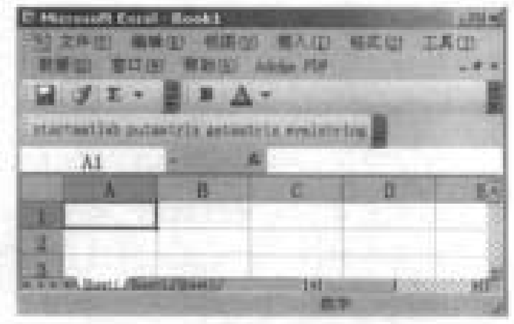


图 31-1 Excel link 工具条

#### 2. Excel link 的启动

在 Excel 启动 MATLAB 的方法有两种：

（1）在 Excel 任意单元格中输入“=MLOpen()”，可启动 MATLAB；

（2）通过选择“工具”菜单中“宏”子菜单的“宏”菜单项，以宏命令 MATLABinit 亦可将 MATLAB 启动起来。

#### 3. Excel link 函数

Excel link 提供了 4 个链接管理函数，其名称和功能如表 31-1 所示。

表 31-1 Excel link 链接管理函数

| 函数名 | 功能 |
|---|---|
| MATLABinit | 初始化 Excel link，并启动 MATLAB |
| MLAutoStart | 自动启动 MATLAB |
| MLClose | 关闭 MATLAB |
| MLOpen | 启动 MATLAB |

Excel link 还提供了 9 个数据管理函数，来实现 Excel 与 MATLAB 之间的数据拷贝和在 Excel 中执行 MATLAB 命令等功能，其名称和功能如表 31-2 所示。

表 31-2 Excel link 数据管理函数

| 函数名 | 功能 |
|---|---|
| matlabfcn | 对给定的 Excel 数据执行 MATLAB 命令 |
| matlabsub | 对给定的 Excel 数据执行 MATLAB 命令，并指定输出位置 |
| MLAppendMatrix | 向 MATLAB 空间添加 Excel 数据表的数据内容 |
| MLDeleteMatrix | 删除 MATLAB 矩阵 |
| MLEvalString | 执行 MATLAB 命令 |
| MLGetMatrix | 向 Excel 数据表写 MATLAB 矩阵的数据内容 |
| MLGetVar | 向 Excel 数据表 VBA 写 MATLAB 矩阵的数据内容 |
| MLPutMatrix | 用 Excel 数据表创建或覆盖 MATLAB 矩阵 |
| MLPutVar | 用 Excel 数据表 VBA 创建或覆盖 MATLAB 矩阵 |

### 31.1.2 Excel link 应用举例

**例 31-1** 实现 Excel 和 MATLAB 中的数据交换。首先在 Excel 中创建一个 3 行 3 列的数据表，然后将其写入到 MATLAB 中，在 MATLAB 中进行运算后，将数据写入到 Excel 中。

基本实现步骤如下：

1. 在 Excel 中创建数据，如图 31-2 所示。
2. 选定数据，选择 putmatrix 按钮，出现如图 31-3 所示的界面。

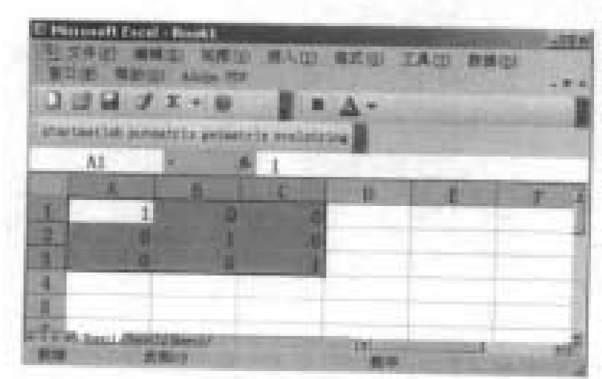


图 31-2 在 Excel 中创建数据

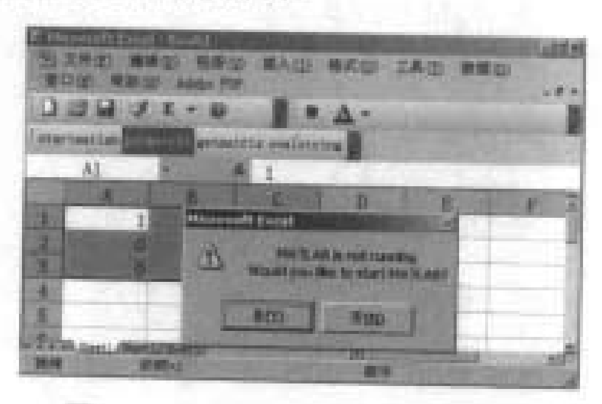


图 31-3 选定数据，选择 putmatrix 按钮

出现警告，没有启动 MATLAB，单击“是”，启动 MATLAB，MATLAB 启动后，出现如图 31-4 所示界面，其中有一个提示窗口，要求输入变量的名称，输入变量名称“t1”后单击“确定”，数据以变量 t1 的形式写入到了 MATLAB 工作空间中。

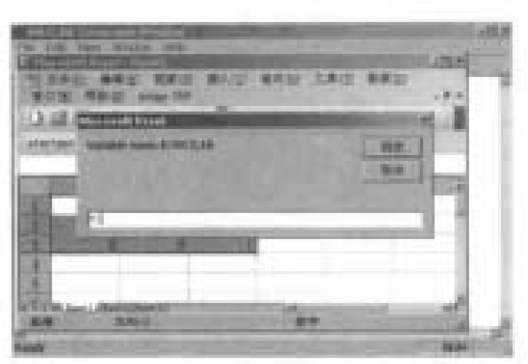


图 31-4 输入变量名称的界面

3. 在 MATLAB 工作空间中，输入变量‘t1’，则 t1 的内容显示出来如图 31-5 所示，显然，它是从 Excel 中传过来的。

4. 在 MATLAB 工作空间中，对变量‘t1’进行乘 2 的运算，并将结果赋给自身，如图 31-6 所示。

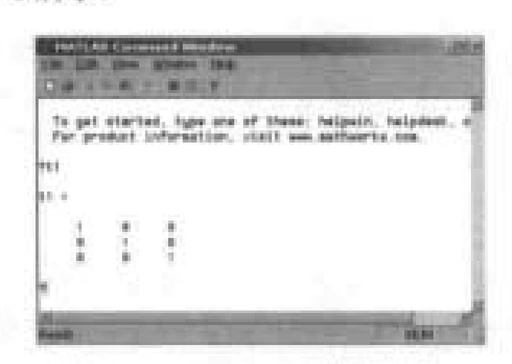


图 31-5 MATLAB 工作空间图

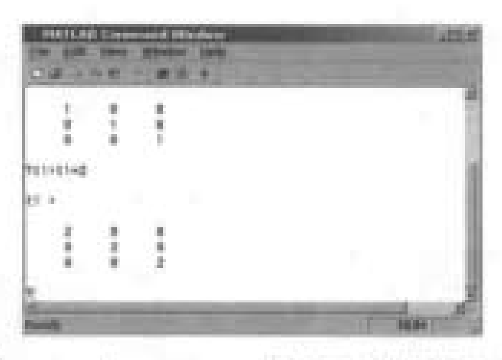


图 31-6 在 MATLAB 工作空间对数据进行运算

5. 在 Excel 中，选定一个区域，单击 getmatrix 按钮，在弹出的窗口中输入变量的名称‘t1’，如图 31-7 所示。

6. 从 MATLAB 中写入的数据在选定区域中显示出来了，如图 31-8 所示。

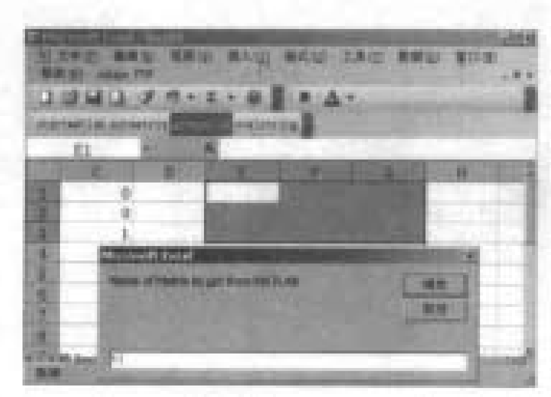


图 31-7 在 Excel 中获取 MATLAB 工作空间的数据

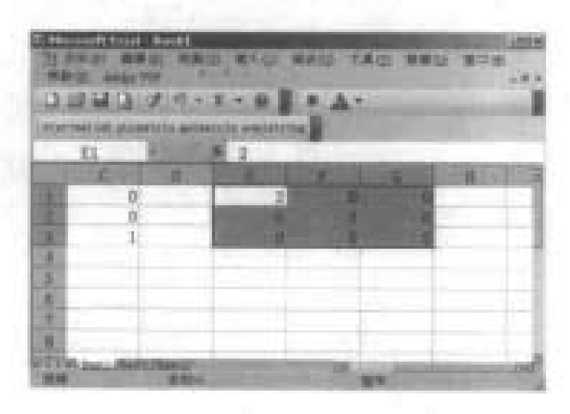


图 31-8 从 MATLAB 工作空间获取的数据



## 31.2 MATLAB 与 VB 接口

Visual Basic 是由 Microsoft 公司开发的在 Windows 平台上的一种十分强大和有生命力的 Windows 编程语言之一，具有易学易用、编程简单、程序集成化程度高及界面可视化，又能实现大多数 Windows 编程目的，因而广受欢迎。然而，VB 提供的数字函数极其有限，因而 VB 在数值计算和图形绘制上显得力不从心。MATLAB 是 MathWorks 公司开发的一种工程计算语言，用这个交互系统可以解决很多工程计算问题，特别是涉及矩阵和矢量形式的问题时，MATLAB 提供了强大的矩阵处理和绘图功能。显然，将二者结合起来，能实现在可视化界面下运用强大的数值计算和图形显示能力，使 VB 编程更加灵活。

### 31.2.1 动态链接库 DLL 方法

Matcom 是一个从 MATLAB 到 C++的编译器，可以节省用户的运算时间和内存要求。Mathtools 公司利用 Matcom4 技术编写了 Mideva 工具软件，可以借用 C++编译器将 MATLAB 下的 M 文件转换为可被 VB 调用的 DLL 或独立可执行文件。VB 中要使用该 DLL，必须在 VB 工程中包含 DLL 的声明文件（即模块文件），同时还要将 DLL 放在工程文件所在的目录下。

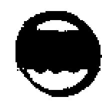
M 文件编译成 DLL 的步骤如下：

（1）启动 Matcom4 的 Mideva，单击菜单 FileCompile，选择要转换的 M 文件应该是作为函数能被其他集成环境调用的 M 文件，例如设文件名为“testfile”。

（2）编译完成后，在对应的 Debug 或 Release 目录下，可以找到一些编译生成的文件。VB 需要用到的文件有两个 testfile.dll（DLL 文件）和 testfile.bas（在 VB 中声明 DLL 的模块文件）。

事实上，testfile.dll 中的函数并不能直接与 VB 进行数据交换，只能通过矩阵数据进行操作。Mathtools 提供了一个单独的 C++库文件 matlib42.dll，以及相应的模块文件 testfile42.bas。matlib42.dll 文件相当于 VB 集成环境与 DLL 之间的代理，它包含的 20 多个矩阵句柄操作函数和 800 多个从 MATLAB 中转换来的矩阵函数。为了在 VB 中能调用这类 DLL，必须将 testfile.dll 和 testfile42.bas 加入工程中并将 testfile.dll 和 testfile42.bas 拷贝到工程所在的目录下。编译后在 VB 中的函数名为 testfile_in_out，其中，in 和 out 分别为函数的输入输出参数的个数。

需要注意的是，在编程时必须调用 mtInitM 来初始化库文件，即请求允许使用转换的 DLL，并调用 maExitM 来结束这种请求。另外，还要使用其他的矩阵句柄函数来分配、访问和释放矩阵句柄。





### 31.2.2 利用 DDE 方式调用 MATLAB 程序

MATLAB 提供了客户/服务器（client/server）功能，利用动态数据交换（DDE）服务功能和 ActiveX 自动化（OLE 自动化）服务功能，可以实现在 VB 程序中对 MATLAB 程序及函数的调用，并且通过 mxArray（MATLAB 中的基本数据类型）与 MATLAB 工作空间交换数据，从而增强 VB 的数值计算能力和数据的可视化能力。

DDE 是在 Windows 环境下支持客户/服务器计算模式的重要技术，两个应用程序之间可以通过交换数据来相互链接。由于 VB 支持 DDE 客户端功能，MATLAB 提供了 DDE 服务功能，因此利用 VB 和 MATLAB 的动态数据交换能力，可以实现在 VB 应用程序中调用 MATLAB 程序。

VB 为用户提供支持通信的控件有：窗体（Form）、多文档窗体（MDI Form）、标签（Label）、文本框（Text Box）和图片框（Picture Box）。Form 相应于 DDE 协议，这些控件提供了完整任务，一次 DDE 链接需要的属性和事件，以及设置必要参数和响应 DDE 过程中的事件。服务器应用程序开发涉及的主要属性如下：

（1）Link Mode 对于一个 Visual Basic 程序来说，作为一个服务器程序和作为一个客户程序时，它们的 Link Mode 属性的取值是不一样的。当一个程序 Visual Basic 作为服务器程序时，Link Mode 属性只是针对 Form 的，它的取值为“1-source”。此时，这个 Form 中的 Picture Box 控件、Text Box 控件和 Label 控件就可以作为客户程序的信息源。

（2）Link Topic 如同 Link Mode 属性一样。对于服务器程序和客户程序来说，属性有不同的意义。Link Topic 属性是针对 Form 的，用户可以将服务器程序中的 Form 的属性设置为任何一个名字，该名字在客户应用程序中使用。

（3）Link Item 指通过一个 DDE 链接传输的数据容器，它是标签（Label）、文本框（Text Box）和图片框（Picture Box）的名称。

### 31.2.3 利用 ActiveX 技术

通过建立 VB 与 MATLAB 间的 ActiveX 自动连接，在 VB 中使用 Set MATLAB= createobject（"matlab.application"）创立 MATLAB 的 ActiveX 对象，其中 matlab.application 是 MATLAB 的 ActiveX 对象在 Windows 注册表中的名称。创建 MATLAB 的 ActiveX 对象后，就可以使用这个对象的各种方法来调用 MATLAB 了。matlab.application 包含以下主要方法：

（1）BSTR Execute (Command as string)

执行 Execute 方法，将调用 MATLAB 执行一条 Command 字符串的 MATLAB 命令，同时以字符串的形式返回命令结果。

（2）GetFullMatrix ([in] BSTR Name, [in] BSTR Workspace, [in] SAFEARRAY (double) pr, [in] SAFEARRAY (double) pi)

将指定的 MATLAB 工作空间中的一个一维或二维数组送到 VB 程序中，Name 指定了





MATLAB 中的矩阵变量名，Workspace 指定了该矩阵所在的工作空间，pr 和 pi 分别是数组的实部和虚部。

（3）PutFullMatrix ([in] BSTR Name，[ink BSTR Workspace， [in] SAFEARRAY (double) pr， [in] SAFEARRAY (double) pi)

将 VB 程序中的一个一维或二维数组传送到指定的 MATLAB 工作空间中。

（4）MinimizeCommandWindow 方法

用于使 MATLAB 命令行窗口最小化。

（5）MaximizeCommandWindow 方法

用于使 MATLAB 命令行窗口最大化。

利用 MATLAB 作为 ActiveX 自动服务器时，VB 应用程序将自动启动 MATLAB 的 ActiveX 自动服务程序，并在程序执行完成后自动关闭 ActiveX 自动服务程序。

## 31.3 MATLAB 与 VC++接口

MATLAB 具有强大的数据处理能力和丰富的工具箱，命令语句功能十分强大，为科学研究、工程设计及众多学科领域提供了一种简洁、高效的编程工具，但其执行效率低，源代码的公开不利于算法和数据的保密，局限于 MATLAB 运行环境而不能用于开发商用软件。

MATLAB 中 MATLAB Compiler 的出现解决了这一难题，它作为单独的工具组件，可以将 M 文件编译并产生 C 或 C++文件，该 C 或 C++文件可以集成到 VC++工程文件中，同时，将 MATLAB C++ Math Library 和 MATLAB 图形库嵌入工程文件，即可产生独立于 MATLAB 环境的可执行程序，从而缩短开发周期。在数值化计算程度高的应用程序开发中，通过二者的混合编程，即采用 VC++完成框架定制和界面开发，利用 MATLAB 强大的矩阵计算和操作以及数值分析功能，完成核心数值计算和算法设计，实现应用程序的快速开发。在 VC++环境中调用 MATLAB 程序的主要方法有使用 MATLAB engine、MEX 文件和 Matcom 三种。

### 31.3.1 使用 MATLAB engine

使用 MATLAB engine（引擎），采用客户机/服务器（client/server）的计算模式。在 VC++中设计程序框架，作为前端客户机，通过调用 MATLAB 引擎与后台 MATLAB 服务器建立连接，实现命令和数据信息的传递。这种方式需要 MATLAB 在后台运行，离不开 MATLAB 环境，不利于软件的开发，但是它可以充分利用 MATLAB 的功能，包括调用工具箱函数和图形函数。

例如，如果用 C/C++语言实现矩阵运算或进行快速傅氏变换算法（FFT），将是非常复杂的，而 MATLAB 提供了功能强大的矩阵运算库函数，使得上述问题的实现变得非常简单。这样，用 VC++作前台界面，MATLAB 作后台分析计算，就可以节约大量开发时间。





MATLAB 提供了一组 MATLAB API（应用程序接口），用户不必关心 MATLAB engine 是如何实现的，只要调用这些 API 即可。正是这些 API 实现应用程序进程之间的数据传递，实现 MATLAB 与 VC++之间的互联。这种方法不要求链接整个 MATLAB，只要求载入 MATLAB 引擎库，节省了大量的系统资源。MATLAB engine 提供的主要 4 个语言调用函数，如表 31-3 所示。

表 31-3 MATLAB 引擎 C 语言函数

| 函数名 | 功能 |
|---|---|
| EngOpen | 开启 MATLAB engine |
| EngClose | 关闭 MATLAB engine |
| EngPutArray | 从应用程序中向 MATLAB engine 发送一个 MATLAB 矩阵 |
| EngGetArray | 从 MATLAB engine 中获得一个 MATLAB 矩阵 |
| EngEvalString | 执行一个 MATLAB 命令 |
| EngOutputBuffer | 创建字符缓冲区以获取 engine 输出 |

### 31.3.2 MEX 文件

利用 MATLAB 的编译器将.m 源文件转化为 C/C++等各种不同类型的源代码，并在此基础上根据应用需要生成 MEX 文件、独立可执行应用程序等文件类型，大大提高程序的运行速度，提高代码执行效率。使用 mcc 命令可以实现.m 文件到 C/C++文件的转化。

MEX 文件是 MATLAB 的外部程序调用接口，用 C 或 Fortran 编写，通过 MATLAB 的 API 函数库对其进行编译，生成一种动态链接函数。MATLAB 可以直接把 MEX 文件视为它的内建函数进行调用，MATLAB 解释器可以自动载入并执行它。MEX 文件主要有以下用途：

（1）对于大量现有的 C 或者 Fortran 程序，可以无须改写成 MATLAB 专用的 m 文件格式而在 MATLAB 中执行；

（2）对于那些 MATLAB 运算速度过慢的算法，可以用 C 或者 Frotran 语言编写，以提高效率。

在使用 MEX 文件方法实现 MATLAB 和 VC++的混合编程之前，需要对 MATLAB 编译器进行配置。

1. 根据外部编译器的类型、软件位置对 m 编译器进行设置。方法是在 MATLAB 命令窗口中执行 mex-setup 命令，然后根据提示选择合适的 C 或 C++编译器，并完成配置。

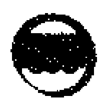
2. 为产生独立外部应用程序进行预配置，并对 MATLAB C 数学函数库进行选择。方法是在 MATLAB 命令窗口中执行 mbuild-setup 命令，然后根据提示进行配置。

3. 在完成 MATLAB 编译器的配置之后，就可以利用该编译器实现和 VC++的互联了。

MEX 文件的编程规则如下：

（1）编制 C++算法程序；

（2）紧跟着定义 mexFunction 函数，mexFunction 的定义法惟一，它只能是如下形式：





void mexFunction(int nlhs，mxArray*plhs[]，int nrhs，const mxArray *prhs[])

其名称和参数类型不许有任何改变，在 mexFunciton 函数中可以调用刚定义好的 C++程序。

为了编写 MEX 文件，MATLAB 提供了一组 MEX 指令。MEX 指令是用来构造 MEX 文件的一些指令，通常以前缀 MEX 开头，如表 31-4 所示。

表 31-4　构造 MEX 文件所用到主要函数列表

| 函数名称 | 功　能 |
|---|---|
| MexFunction | 定义 MEX 文件的接口函数 |
| MexGetArrayPtr | 得到 MATLAB 工作空间中矩阵的指针 |
| MexGetArray | 得到 MATLAB 工作空间中矩阵的一个拷贝 |
| MexPutArray | 保存矩阵 MATLAB 工作空间中 |
| MexCallMATLAB | 调用 MATLAB 指令、M 函数和其他 MEX 文件 |
| MexEvalString | 执行 MATLAB 中的命令 |

表 31-4 中的函数是在 MATLAB 运行时，被调用的 MEX 文件与 MATLAB 之间进行数据交换和控制转移的一些指令，如从 MATLAB 读取或存入数据、执行 MATLAB 的函数、向 MATLAB 输出错误信息等。

### 31.3.3　使用 Matcom 实现 MATLAB 到 C++代码转换

Matcom 是 Mathtools 公司开发的第一个由 MATLAB 到 C++的编译软件开发平台，提供对 MATLAB 程序文件（m 文件）的解释执行和开发环境支持。

Matcom 编译 m 文件，先将 m 文件按照与 Matcom 的 cpp 库的对应关系，翻译为 cpp 源代码，然后用 C 编译器将 cpp 文件编译成相应的 exe 或 dll 文件。

用 Matcom 方式，生成的代码可读性好，支持图形函数，支持 m 文件编译过程中的文件嵌套情况，可脱离 MATLAB 环境；其缺陷是待编译的 m 文件不能涉及 MATLAB 的内部类，而且必须额外地安装 Matcom 软件，故应用范围较窄。

MATLAB 包括数学函数和工具箱函数，Matcom 已经将一般数学函数进行编译，可以遵循 Matcom 语言规则直接在 VC++中使用。要使用工具箱函数，则需要在 Matcom 下编译 MATLAB 的 m 文件，使用 Matcom 的转换过程如图 31-9 所示。

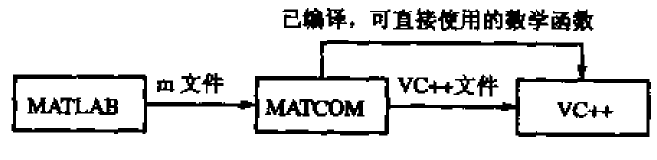
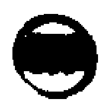



图 31-9　Matcom 转换示意图

上述 3 种方法中，其中通过使用 MATLAB Engine、MEX 方法方法生成的程序必须要求安装 MATLAB 系统，这样程序才能正常运行。而基于 Matcom 方法开发的应用程序则没有这样的要求，它能够以独立执行程序的形式运行，即使在客户没有安装 MATLAB 系统下



也能运行。与基于 MATLAB Engine、MEX 方法生成的应用程序相比，Matcom 方法主要具有以下几个优势：

（1）执行速度快；

（2）内存需求小；

（3）可以发布给没有安装 MATLAB 的用户使用。

Matcom 方法的缺点就是不能利用 MATLAB 中丰富的图形句柄处理函数，但是对于 VC++等开发工具而言，这不是一个很严重的问题。因此，Matcom 方法是实现功能和效率兼顾最好的接口方法。

## 31.4 与 MAT 文件交换数据

MATLAB 与其他编程环境的数据交换是通过 MAT 文件来实现的，MAT 文件是 MATLAB 系统默认的保存文件的格式，这种格式为在不同平台或不同应用程序之间交换 MATLAB 数据提供了一种便利的机制。

为了方便用户在 MATLAB 环境外使用 MAT 文件，MATLAB 提供了一个接口函数库。通过这个接口函数库，用户可以在自己编写的 C 或 Fortran 程序中读写 MAT 文件。

MAT 文件可以存储一个或多个矩阵数据。数据在文件中以顺序格式存储，每个矩阵数据的开始是固定长度的矩阵信息，此信息记录了矩阵的特征信息，如类型、维数等，然后是矩阵的数据，这部分数据所占据的磁盘空间的大小由信息头中反映矩阵大小的决定。与 MAT 文件操作有关的函数列表如表 31-5 所示。

表 31-5 与 MAT 文件操作有关的函数列表

| 函数指令 | 用 途 |
|---|---|
| matOpen | 打开一个 MAT 文件 |
| matClose | 关闭一个 MAT 文件 |
| matGetDir | 从 MAT 文件中得到 MATLAB 所有矩阵名的列表 |
| matGetFp | 得到 C 文件的句柄 |
| matGetArray | 从 MAT 文件中读取 MATLAB 矩阵 |
| matPutArray | 向 MAT 文件中存入 MATLAB 矩阵 |
| matGetNextArray | 从 MAT 文件中读取下一个 MATLAB 矩阵 |
| matDeleteArray | 从 MAT 文件中删除 MATLAB 矩阵 |

下面是一个简单的 C 程序。该程序首先调用 API 例程，将一个一维数组 Areal 转换为 MATLAB 数组，然后调用 MAT 文件子例程，将 MATLAB 数组以 MAT 文件保存。

```
/*定义程序中用到的头文件*/
#include <string.h>
#include "mat. h"
#include "matrix.h"
 static double Areal[6]={10,20,30,40,50,60};  //定义数组 Areal 并给其赋值
 main()
```



```
{
MATFile    *fp;    //定义MAT文件指针
mxArray    * a;    //定义MATLAB数组指针
fp=matOpen("dataout.mat","w");   //打开MAT文件
a=mxCreateDoubleMatrix(3, 2, mxREAL);   //创建mxArray类型的数组
memcpy (mxGetPr (a), Areal, 6*sizeof(double) );   //生成MATLAB数组'a'
mxSetName (a, 'A');  //对数组命名
matPutArray (fp, a) ;  //将数组'a'写入文件'fp'中
matClose (fp);  //关闭打开的MAT文件
mxDestroyArray (a);  //删除已经赋值的指针变量
}
```

## 31.5　小结

本章介绍了MATLAB与Excel，VB，VC++的接口，以及与MAT文件的数据交换，掌握这些方法是熟练应用MATLAB编程接口的基础。

# 第 32 章

# 扩展 MATLAB 和 Java

Java 是一种非常强大的语言，已经成为网络编程的主流语言，而 MATLAB 具有强大的计算功能，把这两种语言结合起来，能大大提高效率。MATLAB 扩展了与 Java 语言的接口。

## 32.1 Java 概述

Java 语言是由 Sun 公司于 1995 年 6 月推出的革命性编程语言，现今广泛使用在网络编程应用方面。它是一种简单的、面向对象的、稳定的、与平台无关的、解释型的、多线程的、动态的语言。

从语言角度来看，Java 具有下列特点。

（1）简单性

Java 继承了 C/C++语法的优点，放弃了其中不常用又容易引起混淆的功能，特别是非面向对象的内容，取消了 C 语言的结构、指针、# define 语句、多重继承、全局变量和函数、GOTO 语句、操作符重载、自动类型转换等等。

（2）面向对象

Java 是一种纯面向对象的语言，具有封装、继承和多态的特性，无全局变量或函数，用于面向对象的现代软件工程。

（3）分布式

Java 包括一个支持 HTTP 和 FTP 等基于 TCP/IP 协议的子库。因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，就像访问本地对象一样。

（4）解释型

Java 语言编写的源代码需要编译成高阶的字节码，它们与机器架构无关。然后，这种



字节码在任何Java的运行环境中由Java虚拟机解释执行，保证了Java的与平台无关性和可移植性。Java语言是解释执行与及时编译技术(JIT)的完美结合，提供了相当高的运算性能。

（5）多态的支持

Java语言非常注重对象形态的转换，因此在编译时期就会做形态转换检查，在执行时期，Java也会做一些形态上的检查。

（6）垃圾收集和异常处理

由于Java垃圾收集器会做自动的垃圾收集(Garbage Collection)，这里的垃圾指一些不会被再使用的对象，所以程序设计者不需费心，内存会被自动地管理，Java本身提供了许多面向对象的异常处理，因此，程序在执行时期所发生的错误，都可以由程序自己来处理。

（7）安全性

Java设计时对系统的安全，特别是网络安全做了周密的考虑。通过字节码验证、内存调用方式、资源使用权限等进行多层次的安全管理。Java被认为是在任何系统上最安全的应用程序之一。

## 32.2　在MATLAB中使用Java

由于Java语言具有如此多的优点，而且已经成为网络编程的主流语言，因此MATLAB在接口中加入了对Java语言的支持。MATLAB适合于数值计算，而Java则适用于网络传输。MATLAB提供了访问Java中类和对象的外部接口支持，MATLAB的这一功能能够方便地将Java类引入到MATLAB运行环境中去，建立这些类的实例，调用这些实例中的函数，以及保存这些实例为以后继续使用，而这些都可通过MATLAB的函数和命令来实现。

如果在安装MATLAB时也安装了Java Virtual Machine (JVM)，就可以通过MATLAB命令调用Java编译器，还可以创建和运行能够创建和访问Java对象的程序。

在命令窗口中输入version –java可以看到MATLAB中使用的JVM的版本，例如在命令窗口输入：

```
>> version -java
%输出为
ans =
Java 1.4.2 with Sun Microsystems Inc. Java HotSpot(TM) Client VM (mixed mode)
```

### 32.2.1　Java接口

MATLAB Java接口是为MATLAB环境中运行和使用Java程序所提供的函数库。通过MATLAB与Java之间的接口能够完成以下功能：

（1）访问Java API的类包，这些类包能够提供一些基本的功能，如输入输出、网络通信等；

（2）访问第三方Java类；

（3）在MATLAB内轻松构造Java对象；

（4）使用 Java 或 MATLAB 语法调用 Java 类对象的方法；

（5）在 MATLAB 和 Java 对象之间进行数据交换。

MATLAB Java 接口主要用于下列用户：

（1）在 MATLAB 环境中访问 Java 类对象的某些功能的用户；

（2）非常熟悉 Java 面向对象编程的用户；

（3）非常熟悉 MATLAB 中的面向对象类，或者熟悉 MATLAB 的 MEX 文件的用户。

MATLAB Java 接口会给这些用户带来很大的方便。例如在面向对象语言编程中，计算的编程比较繁琐时，可以考虑采用 MATLAB 来完成计算功能，然后通过 MATLAB Java 接口来完成与应用程序的通信和交互。

## 32.2.2 MATLAB 中调用 Java

### 1．MATLAB 中调用的 Java 类

按照 Java 类的不同情况，可以将在 MATLAB 中调用 Java 分为以下三类：

（1）Java 内建类（built-in class），对于 Java 本身内建的类库，可以在 MATLAB 中像使用 MATLAB 变量一样方便，例如：

创建一个带有缺省属性的 Frame 对象

frame =Java. awt. Frame('Frame A');

在 MATLAB 下查看 frame 对象结果为：

```
frame=
Java.awt. Frame [frame0, 0, 0, 0x0, invalid, hidden, layout=
Java .awt. BorderLayout, resizable, title= Frame A]
```

（2）自己编写的类（User-Defined class）；

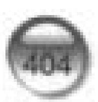
（3）来自第三方（The third-party）的类包，第三方为特定用途开发的类包。

### 2．Java 类的路径

类的路径是 MATLAB 用来定位定义类的一系列文件和目录的说明。当装载某一特定的 Java 类时，MATLAB 在文件和目录中搜索包含类定义的文件。

Java 类路径包含两段路径：静态路径和动态路径。静态路径在每个 MATLAB 任务开始时载入，而且只有在重新启动 MATLAB 后才可以改变；动态路径在使用 MATLAB 功能的任务的任何时候都可以进行载入和修改。MATLAB 总是先搜索静态路径后搜索动态路径。需要注意的是，静态路径下的 Java 类不能与动态路径下的 Java 类存在依赖关系。使用函数 javaclasspath 可以查看这两段路径。例如，在命令窗口输入：

```
>> javaclasspath
%输出为
          STATIC JAVA PATH
     .
     D:\MATLAB7\java\patch
     D:\MATLAB7\java\jar\util.jar
     D:\MATLAB7\java\jar\widgets.jar
```





```
D:\MATLAB7\java\jar\beans.jar
D:\MATLAB7\java\jar\hg.jar
D:\MATLAB7\java\jar\ice.jar
D:\MATLAB7\java\jar\ide.jar
D:\MATLAB7\java\jar\jmi.jar
D:\MATLAB7\java\jar\mde.jar
…（此处为了节约篇幅，用省略号代替）
D:\MATLAB7\java\jarext\xercesImpl.jar
D:\MATLAB7\java\jarext\xml-apis.jar

    DYNAMIC JAVA PATH

    <empty>
```

使用静态路径和动态路径时，通常把更稳定的Java类定义放在静态类路径下，把可能修改的Java类定义放在动态路径下。从静态路径载入比从动态路径载入要快，在动态路径下修改类定义不需要重新启动MATLAB。

为了使自己编写的类库或第三方提供的类库可以被MATLAB调用，必须将自己的类库或第三方提供类库的路径加入classpath.txt文件，例如要使用d: \work\Javaclasses\test.class类库，应该在classpath.txt文件中加入d:\work\Javaclasses，在classpath.txt文件中添加完类库后，重新启动MATLAB，MATLAB会自动调用自己编写的类库或第三方提供的类库，使用它们同使用MATLAB自身变量一样。

修改静态路径需要编写classpath.txt文件，然后重新启动MATLAB。

默认的classpath.txt文件位于MATLAB根目录（matlabroot）的toolbox\local子目录下：[matlabroot '\toolbox\local\classpath.txt']

MATLAB只在启动时读取classpath.txt文件，因此，如果在MATLAB运行时编辑classpath.txt文件或改变.class文件，必须重新启动MATLAB才能使得这些变动有效。

### 3. Java类的表

通常当使用一个Java类时，MATLAB将自动装载该类。任何情况下，运用函数inmem就可以获得一份当前被装载的Java类的表：

[M,X,J]=inmem

该函数返回三个参数，返回值J内包含了目前所有导入的Java类对象。（参数M内是当前装载的M文件，参数X是当前装载的MEX文件）。例如，在命令窗口输入：

```
>> [M,X,J]=inmem
%输出为
M =
    'matlabrc'
    'pathdef'
    …（此处为了节约篇幅，用省略代替）
    'mdbstatus'
    'lasterror'
X =
   Empty cell array: 0-by-1
J =
    'java.util.Locale'
```

```
    'GObject'
    'schema.class'
    'figure'
    'schema.method'
    'java.lang.String'
    'java.lang.CharSequence'
```

MATLAB 命令一般通过完整的类名来引用任何一个 Java 类，其中也包括了类包名，如：

```
java.lang.String
java.util.Enumeration
```

但是一个完整的类名可能会非常长，因为类名还包括类包的名字，使用起来极为麻烦。通常可以首先通过把完整的类名导入到 MATLAB 中，使得可以凭类名称来应用各个类。

导入命令如：

```
import pkg_name.*                    % 导入包中的所有类
import pkg_name1.* pkg_name2.*  % 导入多个包
import class_name                    % 导入一个类
import                               % 显示当前导入列表
L = import                           % 返回当前的导入列表
```

执行 import 命令时，MATLAB 将 import 命令后面的类名添加到导入列表中，可以输入不带参数的 import 命令，查看当前导入列表的内容，这样就可以使用类名而不用加上类包名了。

4. Java 数据类型与 MATLAB 数据类型的转换

从 Java 对象方法返回的 Java 对象，不能被 MATLAB 自动转换为相应的 MATLAB 变量类型，但 MATLAB 提供了如下几个函数，完成相关的转换。

（1）double 函数可以转换任何带有 toDouble 方法，或间接、直接从 Java.lang.Number 继承来的 Java 类对象为 MATLAB 的 double 类型的数据变量，如 double ( Javaobject )。

（2）char 函数可以转换任何 Java.lang.String 对象及 String 数组，或带有 toChar 方法的 Java 类对象为 MATLAB 的 char 数组或单元（cell）数组，如 char ( Javaobject )。

（3）struct 函数或 cell 函数可以转换许多类似 MATLAB struct 或 cell 结构的 Java 类对象。

## 32.3 创建和使用 Java 对象

在 MATLAB 中，可以调用与类名同名的 Java 类构造函数来创建一个 Java 对象，然后使用命令和编程语言来执行对这些对象的操作，也可以把 Java 对象保存为 MAT 文件，以便在以后将其重新载入到 MATLAB 中。

### 32.3.1 创建 Java 类对象

通过调用 Java 类构造函数创建 Java 对象，该对象与类同名，例如在命令窗口输入：



```
>> frame = java.awt.Frame('Frame A');
>> frame
% 输出为
frame =
java.awt.Frame[frame3,0,0,0x0,invalid,hidden,layout=java.awt.BorderLayou
t,title=Frame A,resizable,normal]
```

创建一个名为"Frame A"的Frame实例，该Frame对象的标题为Frame A，类对象的其他属性为默认值。

需要注意的是，在MATLAB中，Java实例是引用形式的，而并非遵循MATLAB分配对象时的拷贝和传递数值的规则。比如，

```
origFrame = java.awt.Frame;
setSize(origFrame, 400, 100);
newFrameRef=origFrame;
```

上述的第三个语句中，变量newFrameRef只是origFrame的第二个引用，而不是该实例的一个拷贝。在接下来的代码中，不管是由MATLAB代码所引起的还是由Java代码引起的，对newFrameRef的任何改变都将同时引起origFrame的改变。比如，改变newFrameRef的大小就会从origFrame中反映出来：

```
>> setSize(newFrameRef, 200, 800);
getSize(origFrame)
%输出为
ans =
java.awt.Dimension[width=200,height=800]
```

## 32.3.2　连接Java对象

就像连接MATLAB中的数据类型一样，可以以相同的方式连接Java对象。使用cat函数或方括号操作符来实现。

**例32-1**　连接相同的Java对象。

**解**：在命令窗口输入：

```
>> point1 = java.awt.Point(24,127);
point2 = java.awt.Point(114,29);
cat(1, point1, point2)  % 使用cat函数
%输出为
ans =
 java.awt.Point[]:
    [java.awt.Point]
    [java.awt.Point]
接着输入，
>> point1 = java.awt.Point(54,27);
point2 = java.awt.Point(14,29);
[point1,point2]  % 使用方括号操作符
%输出为
ans =
 java.awt.Point[]:
    [java.awt.Point]
```

```
    [java.awt.Point]
```

例 32-2 连接不同的 Java 对象。

解：在命令窗口输入：

```
>> byte = java.lang.Byte(127);
integer = java.lang.Integer(52);
double = java.lang.Double(7.8);
[byte; integer; double]  % 使用方括号操作符
 %输出为
ans =
 java.lang.Number[]:
    [   127]
    [    52]
    [7.8000]
```

### 32.3.3 调用 Java 类对象

#### 1. 采用 Java 语法

可以采用下列 Java 语法来调用 Java 实例的函数：

object.method(argl,⋯,argn)

例 32-3 利用 Frame 的 getTitle 和 setTitle 两个函数。

解：在命令窗口输入：

```
>> frame = java.awt.Frame('Frame A');
frame.setTitle('Sample Frame')
title=frame.getTitle
%输出为
title =
 Sample Frame
```

此外，也可以通过 MATLAB 的语法来调用 Java 函数，又如，

```
>> setTitle(frame,'Sample1 Frame')
title=getTitle(frame)
%输出为
title =
Sample1 Frame
```

MATLAB 提供了几个函数来获取正在使用的方法的信息。用户可以获得一个列表，列表的内容是所有已经执行的任何类的方法，这个列表还显示如变量类型等其他的方法信息。如果一个特定的方法在很多已经导入到 MATLAB 中的类中均有，那么可以得到已经执行了这个特定方法的类的一个列表。

#### 2. 采用 methodsview 函数

使用 methodsview，就可以知道某一个方法是否被一个特定的 Java 类或者 MATLAB 类所执行。methodsview 函数的命令格式为：

（1）methodsview packagename.classname；



（2）methodsview classname；

（3）methodsview(object)。

例如，在命令窗口输入 methodsview java.awt.MenuItem，它将列出 java.awt.MenuItem 类的所有方法的信息。

```
>> methodsview java.awt.MenuItem
```

输出如图 32-1 的结果。

图 32-1 java.awt.MenuItem 类的方法

除了使用函数 methodsview 外，通过 MATLAB 的 methods 函数也可以得到有关 Java 类的方法的信息。这个函数的命令格式为：

（1）m = methods('classname')；

（2）m = methods('object')；

（3）m = methods(···, '-full')。

可以看出，methods 有 3 种格式，其中不带“-full”限制符的 methods 将返回类中所有的方法的名字（包括继承来的方法），重载的方法仅列出一个；带"-full"限制符的 methods 将返回一个类中所有方法的列表（包括继承来的方法），列表中有名字、属性、变量表和继承信息，重载的方法分别列出。

### 3. 采用 which 函数

有时用户可能想知道自己正在使用的方法是在哪里定义的，也就是想了解这个方法的归属，可以使用 which 函数。which 函数可以显示一个方法的完整名称，也就是类组件名+类名+方法名（其中的类必须是导入到 MATLAB 中来的）。如果 which 函数带上-all 限制符，which 函数将显示所有拥有这个方法的类。

which 函数对 Java 类对象的操作与对 MATLAB 对象的操作是不一样的。对 MATLAB 对象，which 函数不管 MATLAB 对象是否被导入，都将显示对象的类信息；而对 Java 类对象，which 函数只有当 Java 类对象的类被导入时，才显示对象的类信息。

### 32.3.4 Java 实例

下面通过一个 Java 接口编程的实例，简要介绍 Java 接口的使用。

例子的功能是读 URL。程序首先打开一个 URL 指定的网站，从该网站的一个文件中读取文本，例子构建一个 Java API 类 java.net.URL，这使得程序能够方便地处理 URLs，然后调用 URL 对象的一个方法来打开一个链接。

例子使用 Java I/O 包中的 java.io 类来读取和显示网站上文件的行，然后创建 InputStreamReader 对象来构建一个 BufferedReader 对象，最后调用 BufferedReader 的一个方法来从网站上读取指定数目的行。

例子的主要步骤如下。

（1）调用构造函数 java.net.URL，URL 构造函数只有一个参数，该参数就是需要访问的 URL 的名称，构造函数将检查输入的 URL 是否是有效的格式，代码如下：

url = java.net.URL('http://www.mathworks.com/support/tech-notes/1100/1109.shtml')

（2）打开一个到 URL 的链接，对 URL 的对象 url 调用 openStream 方法，以建立一个该对象命名的网站的链接。此方法返回一个 InputStream 对象给变量 'is' 用来从网站读取字节，代码如下：

is = openStream(url)

（3）建立一个缓冲的流读取，创建读取字符的缓冲的流读取的代码如下：

isr = java.io.InputStreamReader(is)

br = java.io.BufferedReader(isr)

首先调用 java.io.InputStreamReader 的输入流为 'is' 的构造函数，返回 'isr' 对象的变量，用来读取字符；然后调用 java.io.BufferedReader 参数为 'isr' 的构造函数，返回 BufferedReader 对象的 'br' 变量，用来读取字符。缓冲的流读取实现字符、数组和行的高效的读取。

（4）读取和显示文本行，从网站的 HTML 的最初的行读取文本，只显示前面 4 行。BufferedReader 的方法 readLine 从网站读取文本的每一行，代码如下：

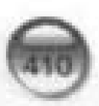
```
for k = 1:108              % 跳过初始的 HTML 格式行
  s = readLine(br);
end

for k = 1:4                % 读文本的前面 4 行
  s = readLine(br)
end
```

## 32.4 Java 与 MATLAB 混合编程

Java 与 MATLAB 混合编程的方法主要有两种：

（1）将 MATLAB 程序文件编译成 DLL 文件，Java 通过 JNI（Java Native Interface）来





调用 DLL 文件，从而间接调用 MATLAB 程序文件，实现混合编程；

（2）由于 Java 不支持 COM，不能象 VC++，VB 一样可以直接通过 COM 组件实现与 MATLAB 的混合编程。开发连接 Java 和 COM 的 Java-COM 桥软件，Java 可以通过 Java-COM 桥调用 COM 组件，实现与 MATLAB 的混合编程，以弥补 Java 不支持 COM 的不足。

下面对第二种混合编程的方法进行介绍。

MATLAB 把.m 文件编译成 COM 组件，同时将 COM 组件的信息登记在注册表中，登记信息中的 ProgID 为该 COM 组件的标识符，jawin 就是通过 ProgID 来调用 COM 组件。

在 Java 编程时通过导入 jawin 包，就可以利用 jawin 包中的类和 ProgID 来初始化 COM、生成对象和调用组件函数。一般步骤如下：

（1）配置 MATLAB，用.m 文件生成相应的 COM 组件；

（2）下载 jawin 压缩包及 JavaSDK 安装包，配置 Java 运行环境及 jawin 包，将 jawindll 复制到系统目录下，在 Java 编程时用 import 导入 jawin.jar；

（3）用引入的包 Ole32CoInitialize()来初始化 COM。

（4）根据 ProgID 生成一个 DispatchPtr（jawin 包中的类）对象，后面将用该对象来调用 COM 组件函数和设置属性；

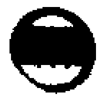
（5）调用 MATLAB 函数输入变量和输出变量的设置。首先生成一个 Integer 对象，用来表示输出参数个数，当没有输出变量时不用生成这个对象，然后输出定义成 Variant.ByrefHolder（jawin 包中的类）对象的变量，并且用空的输出变量的数据类型构造该对象，输入对象为一般的 Object 对象；

（6）用生成的 DispatchPtr 对象的 invoke 函数来调用 MATLAB 函数以及输入、输出变量；

（7）取出输出变量的值，通过强制转换表示输出变量的 Variant.ByrefHolder 对象，通过 getRef()得到的对象到输出变量的数据类型对象。

## 32.5 小结

本章首先简要介绍了 Java，重点讲述了在 MATLAB 中使用 Java、创建和使用 Java 对象以及 MATLAB 与 Java 混合编程。



# 第 33 章 Windows 应用程序集成

MATLAB 的 Windows 应用程序集成是 MATLAB 体系的一个重要功能，MATLAB 通过 COM，DDE，Notebook 等工具与其他软件（例如 VB，VC++，Excel，Word）集成在一起，实现复杂的应用程序。

## 33.1 COM 组件

### 33.1.1 COM 简介

组件对象模型（Component Object Model，简称 COM）是由 Microsoft 公司 1993 年提出的组件标准，现在是微软公司、数据设备公司等所支持的一种软件组件结构标准。它为组件软件和应用程序之间的通信提供了统一的标准，并为组件程序提供了一个面向对象的活动环境。COM 规范定义的组件模型具有以下特性：

（1）面向对象的编程；

（2）组件与开发的工具语言无关；

（3）运行效率高、易扩展、便于使用和管理；

（4）组件的可重用性高。

COM 标准包括规范和实现两大部分：

（1）COM 规范部分定义了组件和组件之间通信的机制，这些规范不依赖于任何特定的语言和操作系统，只要按照该规范，任何语言都可使用 COM 组件；

（2）COM 标准的实现部分是 COM 库，COM 库为 COM 规范的具体实现提供了一些核心服务。这就意味着描述一个对象的可执行代码（.dll 或.exe 文件的代码）可以被其他对象执行。即使两个对象是使用不同语言来编写的，它们也可以用 COM 组件来通信。



COM 不是一种面向对象的语言，而是一种组织软件的方法，是一种协议。同时 COM 是一种客户/服务器程序模式，具有可扩展的体系结构和很高的代码复用率。

COM 为组件提供了编程模型和二进制标准，组件实际上是一些可执行的二进制程序，可以给应用程序、操作系统提供服务。COM 允许组件向其他组件或应用程序展示其功能。通常组件由数据及操作数据的函数组成。应用程序是通过一个或多个相互关联的函数集合来访问组件数据的，这些函数集称为“接口”，“接口”中的函数称为“方法”。

## 33.1.2 MATLAB COM 编译器

### 1. 设置编译环境

MATLAB COM Builder 在编译生成 COM 组件时需要借助于外部的编译器。并不是所有的编译器都生成和 Microsoft 相兼容的 COM 组件，COM Builder 只支持下列编译器：

（1）Borland C++ Builder 4

（2）Borland C++ Builder 5

（3）Borland C++ Builder 6

（4）Microsoft Visual Studio 5.0

（5）Microsoft Visual Studio 6.0

（6）Microsoft Visual Studio .NET

在 MATLAB 安装了 COM Builder（一般在安装 MATLAB 的时候只要选中 COM Builder 就可以了）之后，要先设置 MATLAB COM Builder 所使用的外部编译器，可以通过命令 mbuild-setup 实现。

在 MATLAB 的 Command Windows 输入命令 mbuild -setup，并根据提示选择合适的编译器。在命令窗口中输入：

```
>> mbuild -setup
%输出为
Please choose your compiler for building standalone MATLAB applications:
Would you like mbuild to locate installed compilers [y]/n? y
%提示定位安装的编译器，选择 'y'，回车后将输出计算机上安装的编译器，如下，
Select a compiler:
[1] Lcc C version 2.4 in D:\MATLAB7\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in D:\Program Files\Microsoft
Visual Studio
[0] None
%选择所用的编译器的序号
Compiler: 2
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
Location: D:\Program Files\Microsoft Visual Studio
%选择确认
Are these correct?([y]/n): y
Try to update options file: C:\Documents and Settings\Administrator\
Application Data\MathWorks\MATLAB\R14\compopts.bat
From template:                              D:\MATLAB7\BIN\WIN32\mbuildopts\
```

```
msvc60compp.bat
    Done . . .
    --> "D:\MATLAB7\bin\win32\mwregsvr D:\MATLAB7\bin\win32\mwcomutil.dll"
    DllRegisterServer in D:\MATLAB7\bin\win32\mwcomutil.dll succeeded
    --> "D:\MATLAB7\bin\win32\mwregsvr D:\MATLAB7\bin\win32\mwcommgr.dll"
    DllRegisterServer in D:\MATLAB7\bin\win32\mwcommgr.dll succeeded
```

其中，mwcomutil.dll 和 mwcommgr.dll 是 MATLAB 自动注册的两个动态链接库，这两个 DLL 是 MATLAB COM Builder 生成的 COM 组件的基础，所有生成的 COM 组件都会使用到这两个 DLL。

2. 创建 COM 组件

使用 MATLAB COM Builder 创建 COM 组件的基本步骤如下。

步骤 1：在 MATLAB 下生成所需的.m 文件，需要注意的是，该.m 文件不能是脚本文件，只能是函数文件。

步骤 2：在 MATLAB 的 Command Windows 中输入命令 comtool，启动 COM Builder 的图形用户界面如图 33-1 所示。

步骤 3：使用 File->NewProject…建立新的工程，出现如图 33-2 所示的工程设置窗口。

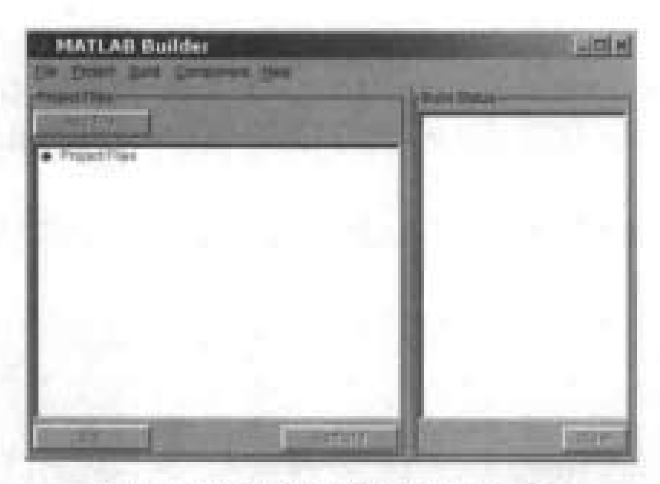



图 33-1　MATLAB COM Builder 主窗口

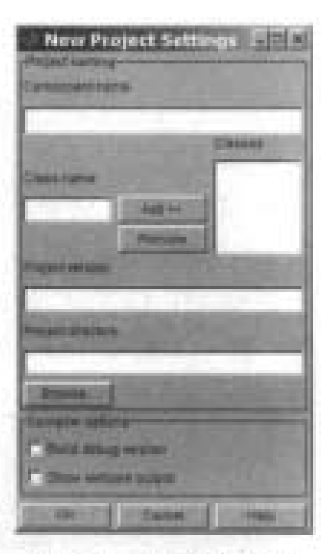



图 33-2　工程设置窗口

步骤 4：进行工程设置窗口的设置。

（1）Component name 中输入所要生成的 COM 组件的名字，例如输入名字 testcom，最后生成的 COM 组件的名字就是这个名字加上版本信息；

（2）通过 Add>>和 Remove 按钮可以方便地为这个 COM 组件添加和删除类；

（3）Project version 是为了方便用户管理自己生成的 COM 组件而设置的，通过这个版本号，用户可以区分不同时期制作的相同名字的 COM 组件；

（4）Project directory 是整个工程存放的目录；

（5）在 Compiler options 中有 Build debug version 和 Show verbose output 两个选项。选中 Build debug version 会生成调试版本的 COM 组件，调试版本的 COM 组件在调试的时候

如果有错误发生，就会有相应的错误提示信息。选中 Show verbose output，可以在编译生成 COM 组件的时候输出详尽的过程信息。

工程创建之后，仍可以通过菜单 Project->Settings…打开此窗口进行修改。一个 COM 组件可以包含许多类，在 Project Setting 里可以实现类的添加。

给一个类添加一个方法如下：先选中所要对其进行操作的类，然后使用菜单 Project->AddFile…或者按钮 Add File 添加现成的（预先编写好的）文件就可以了，项目文件的窗口如图 33-3 所示。

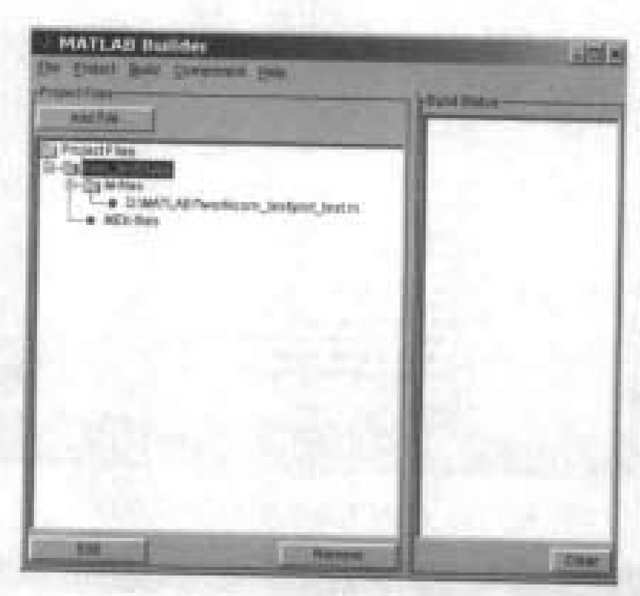


图 33-3 项目文件窗口

步骤 5：所有函数添加完毕之后，就可以使用菜单 Build->COM Object 或者按钮 Build 进行编译了，如图 33-4 所示。

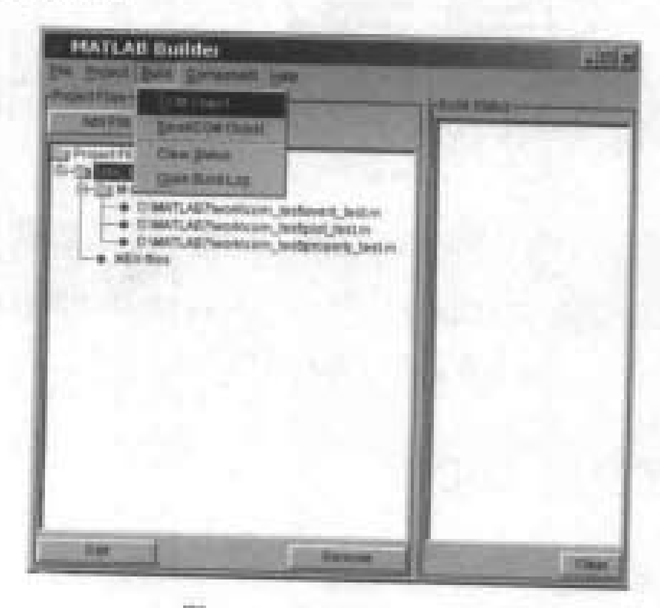


图 33-4 编译 COM 对象

编译过程输出信息如图 33-5 右侧的窗口所示。

最后输出 Standalone DLL build complete 表示 COM 对象编译完成。

编译完毕后会在工程文件夹下生成两个文件夹：一个是 src，里面存放的是一些中间文件；一个是 distrib，里面就是编译而成的文件。在编译完毕之后，MATLAB 会自动组成生成的 component，可以使用菜单 Component->ComponentInfo…将系统注册表里面有关于 dll 的详细信息调出来，看一看有关生成的 component 的信息，如图 33-6 所示。



图 33-5　COM 对象编译过程输出信息

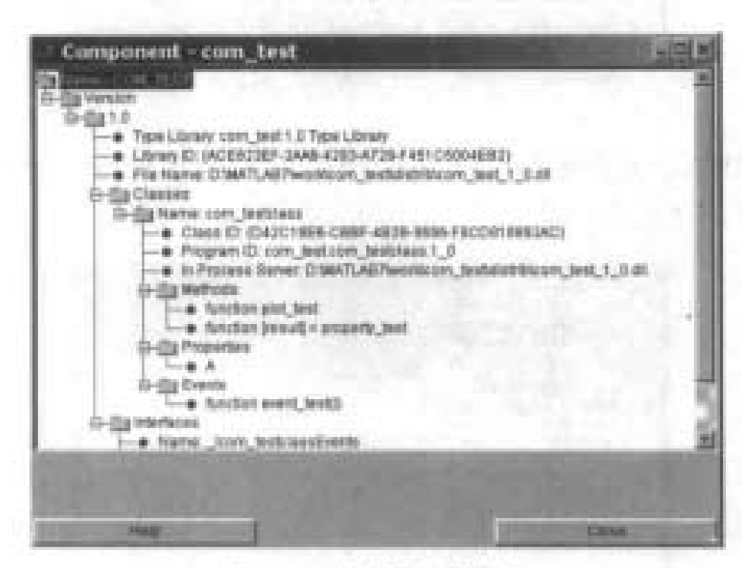



图 33-6　生成的组件信息

## 33.2　动态数据交换（DDE）

### 33.2.1　DDE 基本概念

DDE（Dynamic Data Exchange，动态数据交换）是 Windows 环境提供的一种基于消息

的进程间通信的方法。进程间通信包括进程之间和同步事件之间的数据传递。DDE 使用共享内存来实现进程之间的数据交换，以及使用 DDE 协议获得传递数据的同步。DDE 协议是一组所有的 DDE 应用程序都必须遵循的规则集。

DDE 会话发生在客户应用程序和服务器应用程序之间，DDE 应用程序可以分为客户、服务器、客户/服务器和监视器 4 种类型：

（1）客户应用程序从服务器应用程序请求数据或服务；

（2）服务器应用程序响应客户应用程序的数据或服务请求；

（3）客户/服务器应用程序既是客户应用程序又是服务器应用程序，它既可发出请求又可提供信息；

（4）监视器应用程序用于调试目的。

DDE 应用程序采用三层识别系统，即应用程序名（application）、主题名（topic）和项目名（item）：

（1）应用程序名位于层次结构的顶层，用于指出特定的 DDE 服务器应用程序名；

（2）主题名更深刻地定义了服务器应用程序会话的主题内容，服务器应用程序可支持一个或多个主题名；

（3）项目名更进一步确定了会话的详细内容，每个主题名可拥有一个或多个项目名。

建立 DDE 会话后，客户应用程序和服务器应用程序可通过 3 种链接方式进行数据交换，即冷链接（cold link）、温链接（warm link）和热链接（hot link）：

（1）客户程序传播一条启动触发消息来开始冷链接，则服务器程序向客户程序提供一次数据，当客户还需要服务器提供更多次的数据时，客户程序必须重新传播启动触发消息；

（2）温链接综合了热链接和冷链接的特点，客户只希望被通知数据是否发生了变化，而不一定要立刻得到新的数据，只有当客户知道数据发生了变化并需要获得它时，再启动与冷链接相同的会话；

（3）服务器程序中，已经被访问的数据可能会随着时间的推移而发生变化，在冷链接中，如果客户不传播启动触发消息，则变化了的数据不会传给客户，而在热链接中，服务器会自动将变化了的数据传送给客户。

MATLAB 支持两种链接方式，一种为热链接方式，另一种为温链接方式。

### 33.2.2 MATLAB 中的 DDE

每个应用程序可以是一个 DDE 服务器，它具有惟一的服务器名称，该名称通常是应用程序的可执行的不带任何扩展的文件名。通常使用的 MATLAB 服务的名称是 MATLAB，EXCEL 服务的名称是 Excel。

DDE 使用 Windows 的剪贴板的数据格式对数据进行格式化，以在应用程序间进行数据交换。MATLAB 作为客户端时，它仅仅支持文本格式的数据传输，而当 MATLAB 作为服务器端时，它支持三种格式的数据传输，分别为文本格式、元文件图（Metafilepict）格式和 XLTable 格式，分别描述如下。



（1）文本格式存放的数据以一个以空字符结束字符缓冲，缓冲中以行存放的数据以一个回车符和一个换行符来结束；如果缓冲中存在以列存放的数据，则在数据中以一个 tab 符来结束。MATLAB 通过支持文本格式来获得远程的 Evalstring 命令执行的结果，同时也可以用来请求矩阵数据。此外，矩阵数据也可以使用文本格式发送给 MATLAB。

（2）元文件图格式是一种用来存放图形数据的格式，不同于位图文件通过存放每一个像素点的数据来存放图形的方式，它是通过记录绘制图形过程中所使用的图形命令和函数的方式来描述图形的。简言之，元文件图格式是由一系列的二进制形式的编码图形函数的集合。MATLAB 支持元文件图格式是为了获取一些远程执行的图形命令执行的结果。

（3）XLTable 格式是一种剪贴板支持的 Microsoft Excel 使用的数据格式，使用这种数据格式的主要目的，是方便与 Excel 的数据交换和提高与 Excel 数据交换的效率。XLTable 格式是一个拥有数据头的二进制的缓冲，数据头描述了缓冲中存放的数据。

### 33.2.3 MATLAB 作为 DDE 的服务器端

客户程序可以通过建立 DDE 对话把 MATLAB 作为服务器端进行访问，具体选用何种方法，取决于所使用的应用程序，建立 DDE 对话的方法主要有以下两种：

（1）如果用户使用的应用程序提供了支持 DDE 的函数或者宏，那么用户可以直接使用这些宏和函数来建立与 MATLAB 间的 DDE 对话，典型的应用程序包括 VB，VC++，Word、Excel 等；

（2）如果用户希望通过自己的应用程序建立与 MATLAB 之间的链接，可以通过使用 MATLAB 引擎函数库或直接使用 DDE。

MATLAB 作为一个“Server”时的通信示意图如图 33-7 所示。

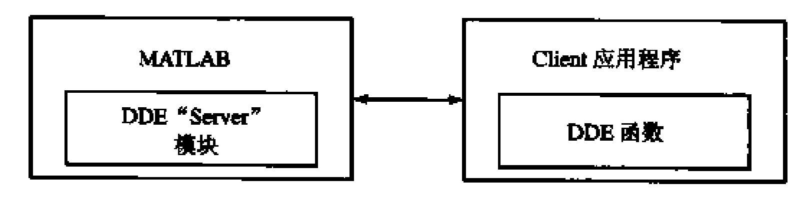



图 33-7　MATLAB 作为服务器时的通信示意图

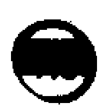
图 33-7 中，在“Client”应用程序中的 DDE 函数与 DDE“Server”模块进行通信，“Client”的 DDE 函数可以由应用程序或 MATLAB Engine Library 提供。

MATLAB 作为服务器使用，它有一个固定的 DDE 名称体系，便于作为服务器被访问，该名称体系包括服务器的名称、主题和项，如图 33-8 所示。

由图 33-8 可见，MATLAB 有两类主题 System 和 Engine，而每类主题又各包含几个不同的项。

System 主题包含了三个项，分别为 SysItems，Format 和 Topics：

（1）SysItems 项提供了一个以 tab 字符结尾的，包含了 System 主题所支持的全部项的列表；



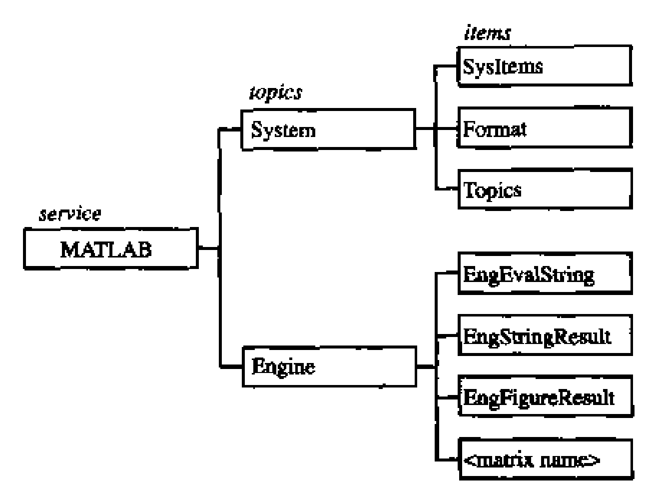



图 33-8　MATLAB DDE 结构图

（2）Format 项提供了一个以 tab 字符结尾的，包含 MATLAB DDE 服务器支持的全部数据格式名的字符串列表，MATLAB 服务器共支持三种类型的数据格式，分别为文本格式、元文件图格式和 XLTable 格式；

（3）Topics 项提供了以 tab 字符结尾的，包含 MATLAB DDE 服务器支持的全部主题名的列表。通过它们，用户可以浏览服务器所提供的主题列表。

Engine 主题就其内容性质而言，分为两类：

（1）第一类把对话内容约定为"客户把指令发至 MATLAB 计算"，所用的描述字是 EngEvalString；

（2）第二类把对话内容约定为"客户向 MATLAB 索取结果"。由于数据性质的不同，又细分为 3 个子类。把对话内容约定为索取"文字结果"，所用的描述字是 EngStringResult；把对话内容约定为"图形"结果，所用的描述字是 EnFigureResult；以 Text 格式或 XLTable 格式，把对话内容约定为索取某个名称的矩阵<matrix name>。为使用方便，Engine 支持的项目属性如列表 33-1 所示。

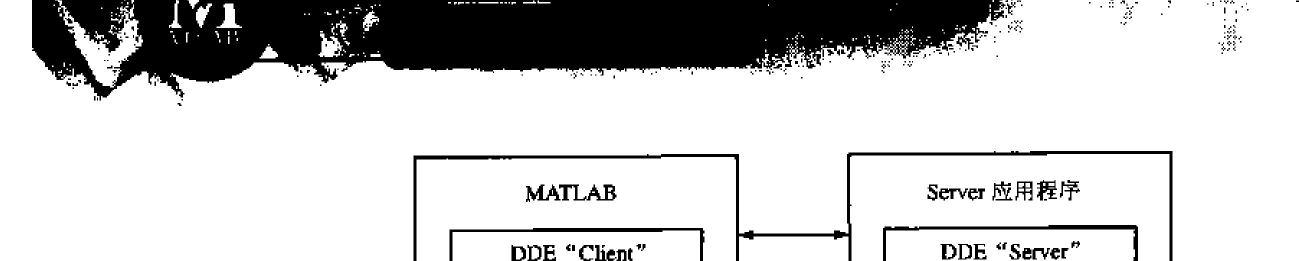
表 33-1　Engine 支持的项目属性

| 项目（Item） | 格式（Format） | 结果（Result） |
|---|---|---|
| EngStringResult | 文本格式 | String |
| EngFigureResult | 文本格式 | 是/否 |
| EngFigureResult | 元文件图格式 | 当前图的元文件 |
| <矩阵名> | 文本格式 | 字符缓冲，表界定的列，CR/LF 界定的行 |
| <矩阵名> | XLTable 格式 | 与 Excel 兼容的二进制数据 |

## 33.2.4　MATLAB 作为 DDE 的客户端

当 MATLAB 作为一个"Client"进行通信时，其示意图如图 33-9 所示。





MATLAB

DDE “Client” 模块

Server 应用程序

DDE “Server” 模块

图 33-9　MATLAB 作为客户端时的通信示意图

“Server”应用程序中的 DDE“Server”模块与 MATLAB 的 DDE“Client”模块进行通信，MATLAB 作为一个“Client”和 Windows 应用程序之间进行动态数据交换时比较方便，只需使用 MATLAB 的 DDE 函数来建立和维持双方的对话。

当 MATLAB 以客户身份建立 DDE 通信时，需要使用 MATLAB 中的 DDE 客户端模块所提供的固有函数，这些函数包括：

（1）DDE 服务初始化函数，channel = ddeinit('service','topic')，初始化 MATLAB 与其他应用程序之间的 DDE 会话；

（2）DDE 链接建立函数，rc = ddeadv(channel,'item','callback','upmtx',format,timeout)，其中，callback 指 item 中元素变化时自动调用的回调函数；upmtx 是一个矩阵，它保存服务器送来的数据，其设置为非空表示上述链接为热链接，否则为温链接；format 是传送的数据格式；timeout 是一个数值，其单位是毫秒，如果这段时间内无法建立链接，此函数调用失败。

（3）DDE 数据请求函数，data = ddereq(channel,'item',format,timeout)，从应用程序请求数据；

（4）DDE 数据发送函数，rc = ddepoke(channel,'item',data,format,timeout)，MATLAB 发送数据给 DDE 应用程序；

（5）DDE 链接释放函数，rc= ddeunadv(channel,'item',format,timeout)，在 MATLAB 与应用程序之间释放链接；

（6）MATLAB 的 DDE 链接终止函数，rc = ddeterm(channel)，在 MATLAB 与应用程序之间终止 DDE 会话；

（7）DDE 执行函数，rc= ddeexec(channel,'command','item',timeout)，发出执行命令至应用程序。

MATLAB 作为 DDE 客户机程序的工作过程如下：

（1）采用 ddeinit 函数与服务器建立对话，建立成功则该函数返回一个通道号，以后的操作均对这个通道号进行；

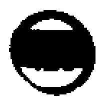
（2）采用 ddeadv 函数请求建立热链接；

（3）采用 ddereq 函数向服务器索要数据，返回值是存有数据的矩阵；

（4）ddepoke 函数向服务器发送数据；

（5）传送结束后，用 ddeterm 函数请求撤消与服务器已建立的热链接。

当程序与服务器建立链接后，进入到等待状态。只要服务器数据发生变化，ddeadv 函数中的 $x$ 矩阵参数就会存储该数据，并执行回调函数 disp($x$)。当然，回调函数在应用的选





取上完全根据设计的需要决定。

## 33.3 Notebook

MathWorks公司开发的MATLAB Notebook成功地将Microsoft Word和MATLAB结合在一起，为文字处理、科学计算和工程设计营造了一个完美的工作环境。这样MATLAB不仅兼具原有的计算能力，而且又增加了Word软件的编辑能力。

### 33.3.1 Notebook基础

MATLAB中的Notebook也可以当作是M-book，它包含了普通的文字内容、MATLAB的指令以及输出的结果。

Notebook可以使用Word环境来处理MATLAB的数值计算，以及所编写的程序输出的结果，方便易用，主要的应用功能有：

（1）文件说明；

（2）MATLAB操作的记录；

（3）项目备忘录；

（4）说明书或者操作手册；

（5）技术报告。

因此可以将操作MATLAB后的结果通过Notebook来做记录，以便用作将来技术转移时的标准文件。简言之，就是可以直接利用Notebook来写作业报告，因为它本身就是利用Word来作文字处理的界面，因此生成的报告就是标准的Word文档。

#### 1. Notebook的安装

首先安装Word，然后启动MATLAB，在其命令窗口输入：

```
>> notebook -setup
%输出为
Welcome to the utility for setting up the MATLAB Notebook for interfacing
MATLAB to Microsoft Word
Choose your version of Microsoft Word:
[1] Microsoft Word 97
[2] Microsoft Word 2000
[3] Microsoft Word 2002 (XP)
[4] Exit, making no changes
%选择Word的版本
Microsoft Word Version: [3]
%输出为
Notebook setup is complete.
```

用户根据所用Word版本，在最后一行提示后面输入对应序号，并按回车键。于是MATLAB会自动寻找winword.exe的安装路径，并在该路径下寻找模板文件normal.dot。如果找到了，则出现提示：

```
Notebook setup is complete.
```

表示 Notebook 安装结束。

2．Notebook 的启动

启动 Notebook 有两种方法：

（1）从 Word 中启动 Notebook；

（2）从 MATLAB 中启动 Notebook。

第二种方法可以直接在 MATLAB 命令窗口中输入：

```
>>notebook
```

来新建 M-book 文件或者输入：

```
notebook<M-book 的文件名>
```

来编辑一个已经存在的 M-book 文件。在输入指令之后，Word 会自动启动。

3．Notebook 的界面

M-book 模板为用户提供了在 Word 环境下使用 MATLAB 的功能。该模板定义了 Word 与 MATLAB 进行通信的宏指令、文档样式和工具栏。当调用该模板时的 Word 界面和通常的 Word 界面主要有两点区别：

（1）在菜单栏中多了一个 Notebook 菜单项，Notebook 的许多操作都可以通过该菜单项的命令来完成；

（2）在"文件"菜单项下多了一个 New M-book 命令项，如果在 M-book 模板下要建立新的 M-book 文档，可以选择该命令。

### 33.3.2 在 Word 中使用 Notebook

MATLAB Notebook 可以在 Word 中随时修改计算命令，随时计算并生成图像返回，使用户能在 Word 环境中方便地使用 MATLAB 的资源。

MATLAB Notebook 的工作方式是：用户在 Word 文档中创建命令，然后送到 MATLAB 的后台中执行，最后将结果返回到 Word 中。

MATLAB Notebook 可以直接在 Microsoft Word 中调用 MATLAB 功能，MATLAB Report Generator 可以将 MATLAB 处理的结果直接生成标准文档。

在 M-book 里输入一般的文字和 MATLAB 指令，其操作方式就和平常使用文字处理的方式一模一样，并没有特别之处。

Notebook 使用输入单元（input cell）来定义 MATLAB 指令，输入单元里可以包括单行或多行的指令，也可以将指令附在文字内容中。

Notebook 中的输入输出单元使用如下：

（1）定义输入单元。首先选中所需命令，然后在 Notebook 菜单项中选择 Define Input Cell 命令，于是被选中的 MATLAB 命令成为输入单元。定义输入单元也可以在选中所需命



令后，直接按组合键 Alt+D。为了执行输入单元，应选择 Notebook 菜单项中的 Evaluate Cell 命令或直接按组合键 Ctrl+Enter。

（2）输入单元执行后产生输出单元。如果输入单元经修改后重新执行，那么新的输出单元将替换原有的输出单元。图形的输出格式则通过 Notebook 菜单中的 Notebook Options 来设置。

例 33-1 在 M-book 文档中定义输入单元，要求产生一个 2 行 2 列的矩阵，并求其平方。

**解**：操作步骤如下：

（1）在文档中输入 MATLAB 命令：

```
a=[1,2;3,2];
b=a^2;
```

（2）选中命令行，在 Notebook 菜单项中选 Define Input Cell 命令或直接按组合键 Alt+D，于是命令行就变成了“绿色”的输入单元。

（3）若要把输入单元送去执行，则可用 Notebook 菜单项中的 Evaluate Cell 命令或直接按组合键 Ctrl+Enter，执行后产生“蓝色”的输出单元。

输入单元的定义与执行也可以同时进行。先选中 MATLAB 命令，然后从 Notebook 菜单项中选择 Evaluate Cell 命令或直接按组合键 Ctrl+Enter，不但使被选中的命令成为输入单元，而且送去执行，产生输出单元。

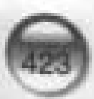
在 MATLAB 命令窗口中输入：

```
>> a
%输出为
a =
     1     2
     3     2
>> b
%输出为
b =
     7     6
     9    10
```

以上简单地谈到了 MATLAB 与 Word 的链接使用，当然 MATLAB Notebook 的功能远不止这些，如还有如何进行分区计算，如何进行循环计算等等，利用 ActiveX 技术还可以制作现场输入参数，由于这些都要利用太多的 ActiveX 以及 MATLAB DDE 技术，在此就不细讲，有兴趣的读者可参阅其他书籍。

## 33.4 小结

本书介绍了 MATLAB 与 Windows 应用程序集成的相关内容，讲述了 COM，DDE，Notebook 等集成工具及其使用方法。



# 第 34 章 Simulink 交互式仿真集成环境

1990 年，MathWorks 软件公司为 MATLAB 提供了新的系统模型化图形输入与仿真工具，并命名为 SIMULAB，该工具很快就在工程界获得了广泛的认可，使仿真软件进入了模型化图形组态阶段，但因其名字与当时比较著名的软件 SIMULA 类似，所以 1992 年正式将该软件更名为 Simulink。

Simulink 的出现为系统仿真与设计带来了福音。顾名思义，该软件有两个主要功能：Simu（仿真）和 Link（连接），即该软件可以利用鼠标在模型窗口上绘制出所需要的仿真系统模型，然后利用 Simulink 提供的功能来对系统进行仿真和分析。

## 34.1 Simulink 的使用

Simulink 是 MATLAB 软件的扩展，是实现动态系统建模和仿真的一个软件包。它与 MATLAB 语言的主要区别在于，其与用户交互接口是基于 Windows 的模型化图形输入，使用户可以把更多的精力投入到系统模型的构建，而非语言的编程上。

所谓模型化图形输入是指 Simulink 提供了一些按功能分类的基本的系统模块，用户只需要知道这些模块的输入输出及模块的功能，而不必考察模块内部是如何实现的，通过对这些基本模块的调用，再将它们连接起来就可以构成所需要的系统模型（以.mdl 文件进行存取），进而进行仿真与分析。

Simulink 的最新版本是 Simulink 5.0（包含在 MATLAB 7.0 里），MATLAB 6.0 里的版本为 4.0 版，它们的基本功能相差不大，一些主要的变化在第二章已做介绍。



### 34.1.1 Simulink 启动

Simulink 的启动有两种方式，一种是启动 MATLAB 后，单击 MATLAB 主窗口的快捷按钮▣，打开 Simulink Library Browser 窗口，如图 34-1 所示。

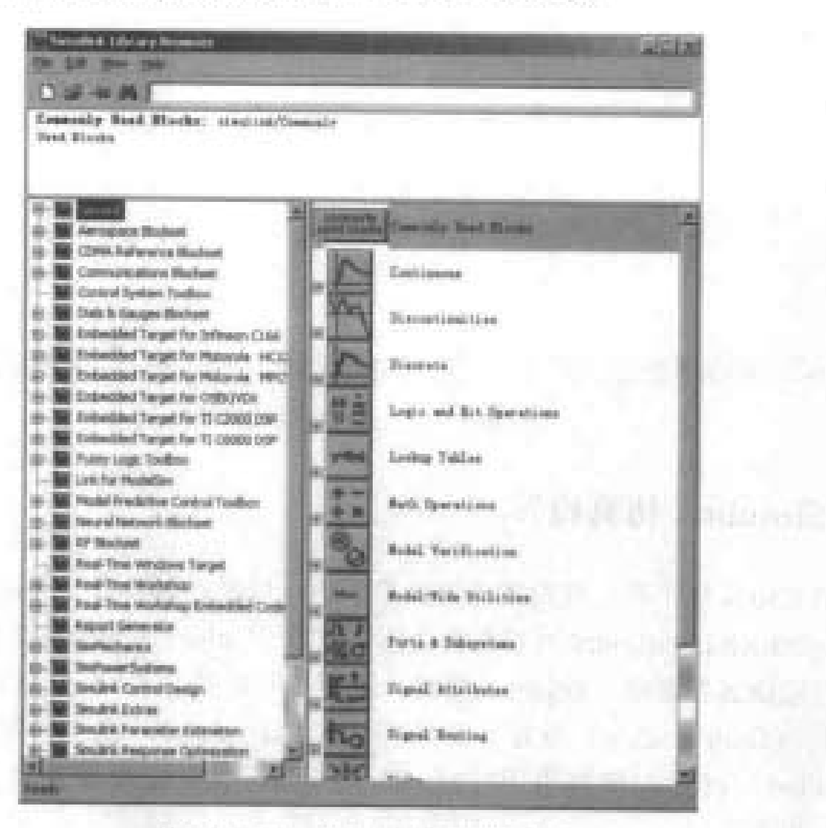


图 34-1 Simulink 模块库浏览界面

另一种是在 MATLAB 命令窗口中输入 simulink，结果是在桌面上出现一个称为 Simulink Library Browser 的窗口，在这个窗口中列出了按功能分类的各种模块的名称。

在 MATLAB 命令窗口中输入 simulink3，结果是在桌面上出现一个用图标形式显示的 Library :simulink3 的 Simulink 模块库窗口，如图 34-2 所示。这两种模块库窗口界面只是不同的显示形式，用户可以根据自己喜好进行选用，一般说来第二种窗口直观、形象，易于初学者，但使用时会打开太多的子窗口。

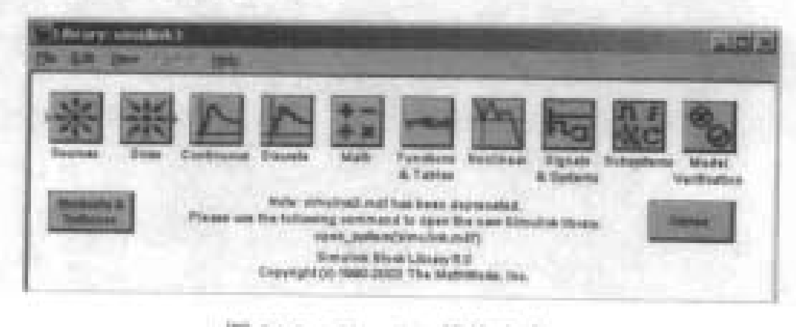


图 34-2 Simulink 模块库窗口

Simulink 启动后，便可打开如图 34-3 所示的 Simulink 的仿真编辑窗口，用户可以开始编辑自己的仿真程序了。

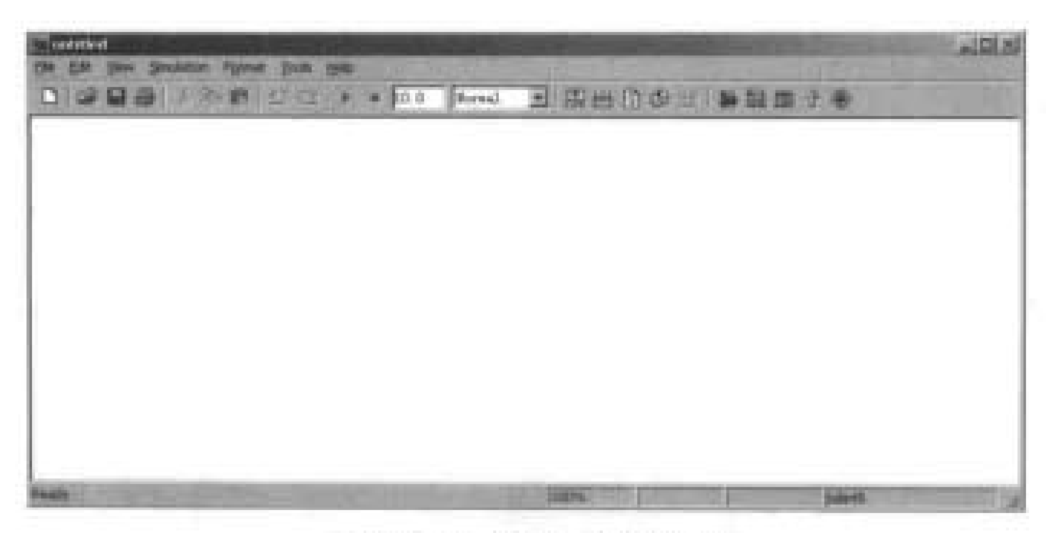


图 34-3　Simulink 仿真编辑窗口

## 34.1.2　Simulink 仿真设置

在编辑完仿真程序后，应设置仿真操作参数，以便进行仿真。单击 Simulation 菜单下面的 Configuration Parameters 项或者直接按快捷键“Ctrl+E”，便弹出如图 34-4 所示的设置界面，它包括仿真器参数（Solver）设置、工作空间数据导入/导出（Data Import/Export）设置、优化（Optimization）设置、诊断参数（Diagnostics）设置、硬件实现（Hardware Implementation）设置、模型引用（Model Referencing）设置和实时工作间（Real-Time Workshop）设置。

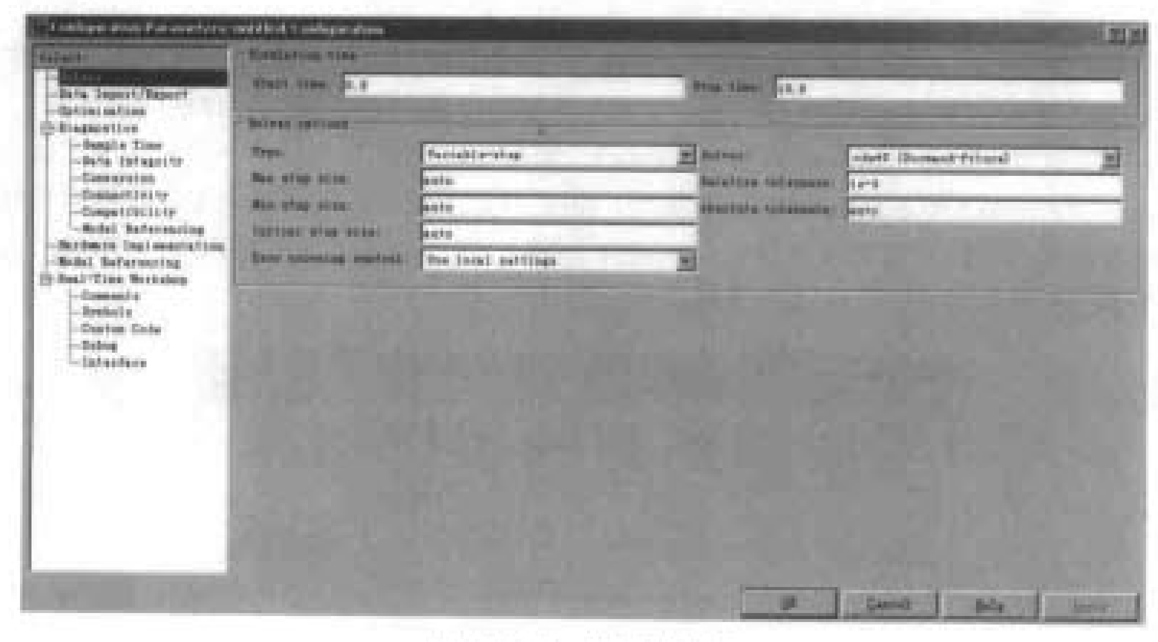


图 34-4　Simulink 设置窗口



### 1．仿真器参数（Solver）设置

仿真器设置的界面如图 34-5 所示，它可用于仿真开始时间、仿真结束的时间、选择解法器及输出项等的选择。

图 34-5　仿真器参数设置窗口

（1）仿真时间

这里所指的时间概念与真实的时间并不一样，只是计算机仿真中对时间的一种表示，比如 10 秒的仿真时间，如果采样步长定为 0.1，则需要执行 100 步，若把步长减小，则采样点数增加，那么实际的执行时间就会增加。一般仿真开始时间设为 0，而结束时间则视不同的情况选择。总的说来，执行一次仿真要耗费的时间依赖很多因素，包括模型的复杂程度、解法器及其步长的选择、计算机时钟的速度等等。

（2）仿真步长模式

用户在 Type 后面的第一个下拉选项框中指定仿真的步长选取方式，如图 34-6 所示，可供选择的有 Variable-step（变步长）和 Fixed-step（固定步长）方式。选择变步长模式则可以在仿真过程中改变步长，提供误差控制和过零检测选择。固定步长模式则可以在仿真过程中提供固定的步长，不提供误差控制和过零检测。用户还可以在第二个下拉选项框中选择对应模式下仿真所采用的算法。




图 34-6　仿真类型设置窗口

用户在 Solver 后面的下拉选项中选择变步长模式解法器，如图 34-7 所示。变步长模式解法器有：ode45，ode23，ode113，ode15s，ode23s，ode23t，ode23tb 和 discrete，下面简要概述一下这些解法器的含义。

① ode45：默认值，表示四/五阶龙格-库塔法，适用于大多数连续或离散系统，但不适用于刚性（stiff）系统。它是单步解法器，即在计算 $y(t_n)$ 时，仅需要最近处理时刻的结

果 $y(t_{n-1})$ 。一般来说，对一个仿真问题最好是首先试试 ode45。

图 34-7 解法器参数设置窗口

② ode23：表示二/三阶龙格-库塔法，它在误差限要求不高，求解的问题不太难的情况下，可能会比 ode45 更有效。它也是一个单步解法器。

③ ode113：表示一种阶数可变的解法器，它在误差容许要求严格的情况下通常比 ode45 有效。ode113 是一种多步解法器，即在计算当前时刻输出时，需要以前多个时刻的解。

④ ode15s：表示一种基于数字微分公式的解法器（NDFs），它也是一种多步解法器，适用于刚性系统。当用户估计要解决的问题是比较困难的，或者不能使用 ode45，或者即使使用效果也不好时，就可以用 ode15s。

⑤ ode23s：表示一种单步解法器，专门应用于刚性系统，在弱误差允许下的效果好于 ode15s。它能解决某些 ode15s 所不能有效解决的 stiff 问题。

⑥ ode23t：表示梯形规则的一种自由插值实现。这种解法器适用于求解适度 stiff 的问题，而用户又需要一个无数字振荡的解法器的情况。

⑦ ode23tb：表示 TR-BDF2 的一种实现，TR-BDF2 是具有两个阶段的隐式龙格-库塔公式。

⑧ discrete：当 Simulink 检查到模型没有连续状态时使用它。

固定步长模式解法器有：ode5，ode4，ode3，ode2，ode1 和 discrete。

① ode5：默认值，是 ode45 的固定步长版本，适用于大多数连续或离散系统，不适用于刚性系统。

② ode4：表示四阶龙格－库塔法，具有一定的计算精度。

③ ode3：表示固定步长的二/三阶龙格－库塔法。

④ ode2：表示改进的欧拉法。

⑤ ode1：表示欧拉法。

⑥ discrete：表示一种实现积分的固定步长解法器，它适合于离散无连续状态的系统。

（3）步长参数

对于变步长模式，用户可以设置最大的和推荐的初始步长参数，默认情况下，步长自动确定，它用 auto 值表示。

① Maximum step size（最大步长参数）：它决定解法器能够使用的最大时间步长，默认值为“仿真时间/50”，即整个仿真过程中至少取 50 个取样点，但这样的取法对于仿真时间较长的系统则可能带来取样点过于稀疏，而使仿真结果失真。一般建议对于仿真时间不

超过 15s 的采用默认值即可，对于超过 15s 的仿真过程每秒至少保证 5 个采样点，对于超过 100s 的，每秒至少保证 3 个采样点。

② Initial step size（初始步长参数）：一般建议使用“auto”默认值。

（4）仿真精度定义（对于变步长模式）

① Relative tolerance（相对误差）：它是指误差相对于状态的值，是一个百分比，默认值为 1e-3，表示状态的计算值要精确到 0.1%。

② Absolute tolerance（绝对误差）：表示误差值的门限，或者是说在状态值为零的情况下，可以接受的误差。如果它被设成了 auto，那么 Simulink 为每一个状态设置初始绝对误差为 1e-6。

（5）Mode（固定步长模式选择）

① Multitasking：选择这种模式时，当 Simulink 检测到模块间非法的采样速率转换，它会给出错误提示。所谓的非法采样速率转换，指两个工作在不同采样速率的模块之间的直接连接。在实时多任务系统中，如果任务之间存在非法采样速率转换，那么就有可能出现一个模块的输出在另一个模块需要时却无法利用的情况。通过检查这种转换，Multitasking 将有助于用户建立一个符合现实的多任务系统的有效模型。使用速率转换模块，可以减少模型中的非法速率转换。Simulink 提供了两个这样的模块：unit delay 模块和 zero-order hold 模块。对于从慢速率到快速率的非法转换，可以在慢输出端口和快输入端口插入一个单位延时 unit delay 模块。而对于快速率到慢速率的转换，则可以插入一个零阶采样保持器 zero-order hold。

② Singletasking：这种模式不检查模块间的速率转换，它在建立单任务系统模型时非常有用，在这种系统就不存在任务同步问题。

③ Auto：选择这种模式时，Simulink 会根据模型中模块的采样速率是否一致，自动决定切换到 Multitasking 和 Singletasking。

（6）输出选项

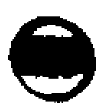
① Refine output：这个选项可以理解成精细输出，其意义是在仿真输出太稀松时，Simulink 会产生额外的精细输出，这一点就像插值处理一样。用户可以在 refine factor 设置仿真时间步之间插入的输出点数。要产生更光滑的输出曲线，改变精细因子比减小仿真步长更有效。精细输出只能在变步长模式中才能使用，并且在 ode45 效果最好。

② Produce additional output：它允许用户直接指定产生输出的时间点。一旦选择了该项，则在它的右边出现一个 output times 编辑框，在这里用户指定额外的仿真输出点，它既可以是一个时间向量，也可以是表达式。与精细因子相比，这个选项会改变仿真的步长。

③ Produce specified output only：它的意思是让 Simulink 只在指定的时间点上产生输出。为此，解法器要调整仿真步长，使之和指定的时间点重合。这个选项在比较不同的仿真时，可以确保它们在相同的时间输出。

### 2. 工作空间数据导入/导出（Data Import/Export）设置

工作空间数据导入/导出设置的界面如图 34-8 所示，它主要用于在 Simulink 与 MATLAB



工作空间交换数值时有关选项设置，包括 Load from workspace，Save to workspace 和 Save option 3 个选择项。

图 34-8　工作空间数据导入/导出设置窗口

（1）Load from workspace：选中前面的复选框，即可从 MATLAB 工作空间获取时间和输入变量，一般时间变量定义为 $t$，输入变量定义为 $u$。Initial state 用来定义从 MATLAB 工作空间获得的状态初始值的变量名。

（2）Save to workspace：用来设置存在 MATLAB 工作空间的变量类型和变量名，选中变量类型前的复选框使相应的变量有效。一般存往工作空间的变量包括输出时间向量（Time）、状态向量（States）和输出变量（Output）。Final states 用来定义将系统稳态值存往工作空间时所用的变量名。

（3）Save options：用来设置存往工作空间的有关选项。Limit data points to last 用来设定 Simulink 仿真结果最终可存往 MATLAB 工作空间的变量的规模，对于向量而言即其维数，对于矩阵而言即其秩；Decimation 设定了一个亚采样因子，它的默认值为 1，也就是对每一个仿真时间点产生值都保存，而若为 2，则是每隔一个仿真时刻才保存一个值。Format 用来说明返回数据的格式，包括矩阵 matrix、结构 struct 及带时间的结构 struct with time。

### 3. 诊断参数（Diagnostics）设置

诊断参数设置的窗口如图 34-9 所示，它包括采样时间（Sample Time）、数据完整性（Data Integrity）、转换（Conversion）、连接（Connectivity）、兼容性（Compatibility）和模型引用（Model Referencing）这几个子项的诊断。用户可以设置当 Simulink 检查到这些子项事件时应做的处理，主要包括是否进行一致性检验、是否禁用过零检测、是否禁止复用缓存、是否进行不同版本的 Simulink 的检验等几项。

### 4. 实时工作间（real-time workshop）设置

仿真参数设置窗口还包括 real-time workshop 页，如图 34-10 所示，主要用于与 C 语言编辑器的交换，通过它可以直接从 Simulink 模型生成代码，并且自动建立可以在不同环境下运行的程序，这些环境包括实时系统和单机仿真。




图 34-9 诊断参数设置窗口

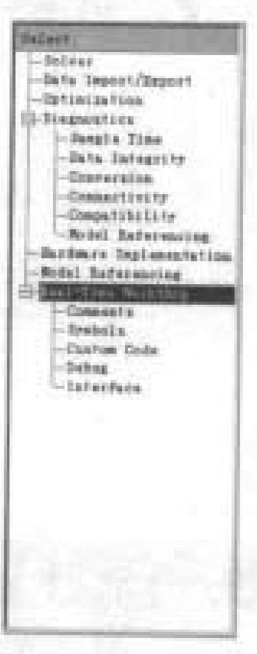



图 34-10 实时工场设置窗口

### 5. 其他设置

仿真参数设置窗口还包括优化（Optimization）、硬件实现（Hardware Implementation）和模型引用（Model Referencing）设置。

Optimization 设置窗口如图 34-11 所示，用于设置模拟仿真的优化参数。

Hardware Implementation 设置窗口如图 34-12 所示，包括 Microprocessor 和 Unconstrained Integer size 两个选项，用于设置仿真硬件特性。

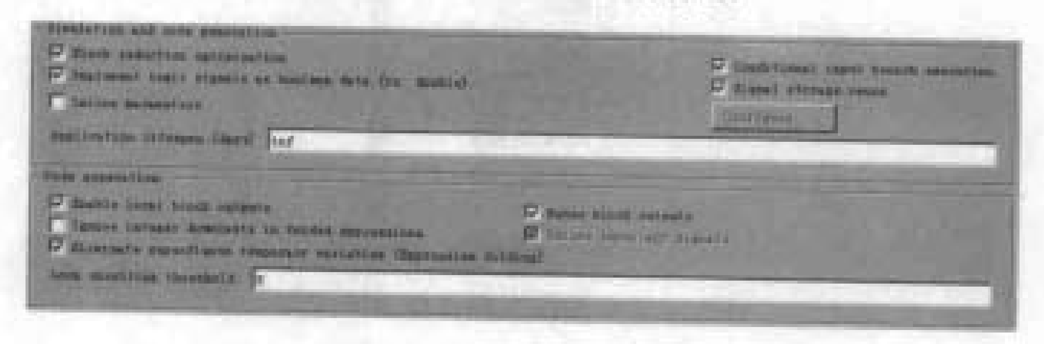


图 34-11 Optimization 设置窗口

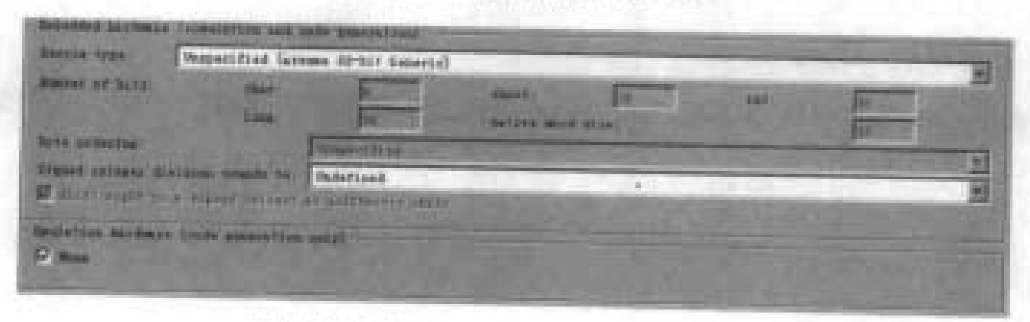
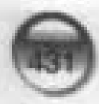


图 34-12 Hardware Implementation 设置窗口



Model Referencing 设置窗口如图 34-13 所示，用于设置模型引用的有关参数。



图 34-13　Model Referencing 设置窗口

### 34.1.3　Simulink 模块库简介

在进行系统动态仿真之前，应绘制仿真系统框图，并确定仿真所需用的参数。Simulink 模块库包含有大部分常用的建立系统框图的模块，如图 34-14 所示。下面简要介绍常用模块。

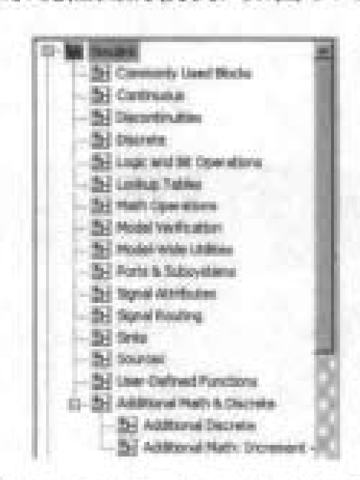



图 34-14　Simulink 模块浏览窗口

#### 1．连续模块（Continuous）

在 Simulink 基本模块中选择 Continuous 后，点击便看到如图 34-15 所示的连续模块，它包括以下子模块：

（1）Derivative 输入信号微分；

（2）Integrator 输入信号积分；

（3）State-Space 状态空间系统模型；

（4）Transfer-Fcn 传递函数模型；



（5）Transport Delay 输入信号延时一个固定时间再输出；

（6）Variable Transport Delay 输入信号延时一个可变时间再输出；

（7）Zero-Pole 零极点模型。

### 2．非连续模块（Discontinuous）

在 Simulink 基本模块中选择 Discontinuous 后，点击便看到如图 34-16 所示的非连续模块，它包括以下子模块：

（1）Backlash 间隙非线性；

（2）Coulomb&Viscous Friction 库仑和粘度摩擦非线性；

（3）Dead Zone 死区非线性；

（4）Dead Zone Dynamic 动态死区非线性；

（5）Hit Crossing 冲击非线性；

（6）Quantizer 量化非线性；

（7）Rate Limiter 静态限制信号的变化速率；

（8）Rate Limiter Dynamic 动态限制信号的变化速率；

（9）Relay 滞环比较器，限制输出值在某一范围内变化；

（10）Saturation 饱和输出，让输出超过某一值时能够饱和；

（11）Saturation Dynamic 动态饱和输出。

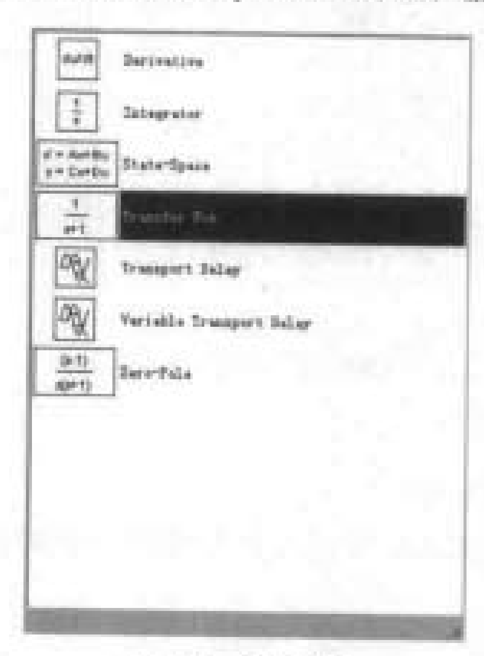


图 34-15　连续模块

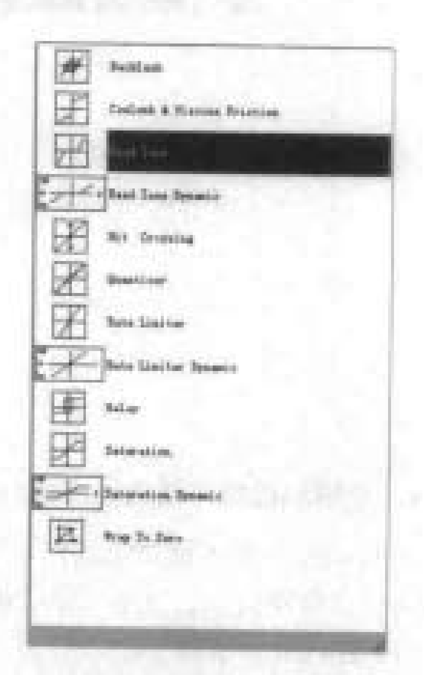


图 34-16　非连续模块

### 3．离散模块（Discrete）

在 Simulink 基本模块中选择 Discrete 后，点击便看到如图 34-17 所示的离散模块，它包括以下子模块：

（1）Difference 差分环节；

（2）Discrete Derivative 离散微分环节；

（3）Discrete Filter 离散滤波器；

（4）Discrete State-Space 离散状态空间系统模型；

（5）Discrete Transfer-Fcn 离散传递函数模型；

（6）Discrete Zero-Pole 以零极点表示的离散传递函数模型；

（7）Discrete-time Integrator 离散时间积分器；

（8）First-Order Hold 一阶保持器；

（9）Integer Delay 整数被延迟；

（10）Memory 输出本模块上一步的输入值；

（11）Tapped Delay 延迟；

（12）Transfer Fcn First Order 离散一阶传递函数；

（13）Transfer Fcn Lead or Lag 传递函数；

（14）Transfer Fcn Real Zero 离散零点传递函数；

（15）Unit Delay 一个采样周期的延时；

（16）Weighted Moving Average 权值移动平均模型；

（17）Zero-Order Hold 零阶保持器。




图 34-17　离散模块

### 4．逻辑和位操作模块（Logic and Bit Operations）

在 Simulink 基本模块中选择 Logic and Bit Operations 后，点击便看到如图 34-18 所示的逻辑和位操作模块，它包括以下子模块：

（1）Bit Clear 位清零；

（2）Bit Set 位置位；

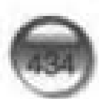
（3）Bitwise Operator 逐位操作；

（4）Combinatorial Logic 组合逻辑；

（5）Compare To Constant 和常量比较；

（6）Compare To Zero 和零比较；

（7）Detect Change 检测跳变；





（8）Detect Decrease 检测递减；

（9）Detect Fall Negative 检测负下降沿；

（10）Detect Fall Nonpositive 检测非负下降沿；

（11）Detect Increase 检测递增；

（12）Detect Rise Nonnegative 检测非负上升沿；

（13）Detect Rise Positive 检测正上升沿；

（14）Extract Bits 提取位；

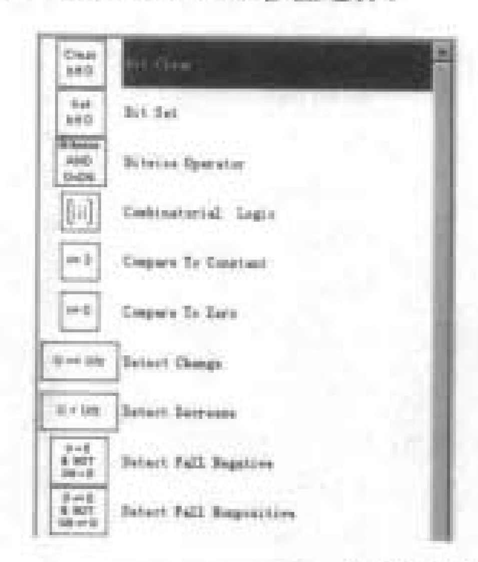
（15）Interval Test 检测开区间；

（16）Interval Test Dynamic 动态检测开区间；

（17）Logical Operator 逻辑操作符；

（18）Relational Operator 关系操作符；

（19）Shift Arithmetic 移位运算。



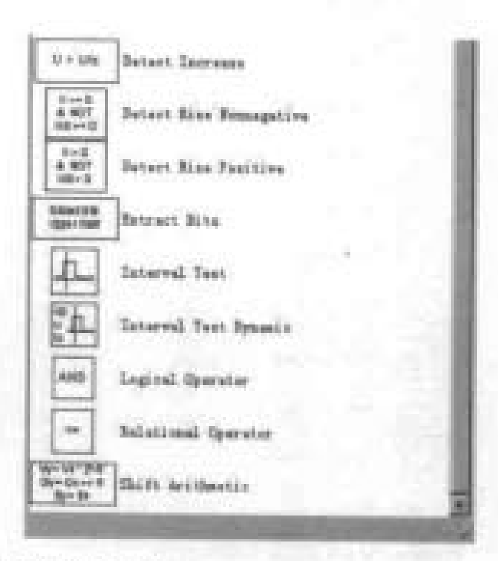


图 34-18　逻辑和位操作模块

图 34-19　查找表模块

### 5．查找表模块（Lookup Table）

在 Simulink 基本模块中选择 Lookup Table 后，点击便看到如图 34-19 所示的查找表模块，它包括以下子模块：

（1）Cosine　余弦函数查询表；

（2）Direct Lookup Table (n-D)　*n* 个输入信号的查询表（直接匹配）；

（3）Interpolation (n-D) using PreLookup　*n* 个输入信号的预插值；

（4）Lookup Table　输入信号的查询表（线性峰值匹配）；

（5）Lookup Table(2-D)　两维输入信号的查询表（线性峰值匹配）；

（6）Lookup Table(n-D)　*n* 维输入信号的查询表（线性峰值匹配）；
（7）Lookup Table Dynamic　动态查询表；
（8）PreLookup Index Search 预查询索引搜索；
（9）Sine 正弦函数查询表。

### 6．数学模块（Math Operations）

在 Simulink 基本模块中选择 Math Operations 后，点击便看到如图 34-20 所示的数学模块，它包括以下子模块：

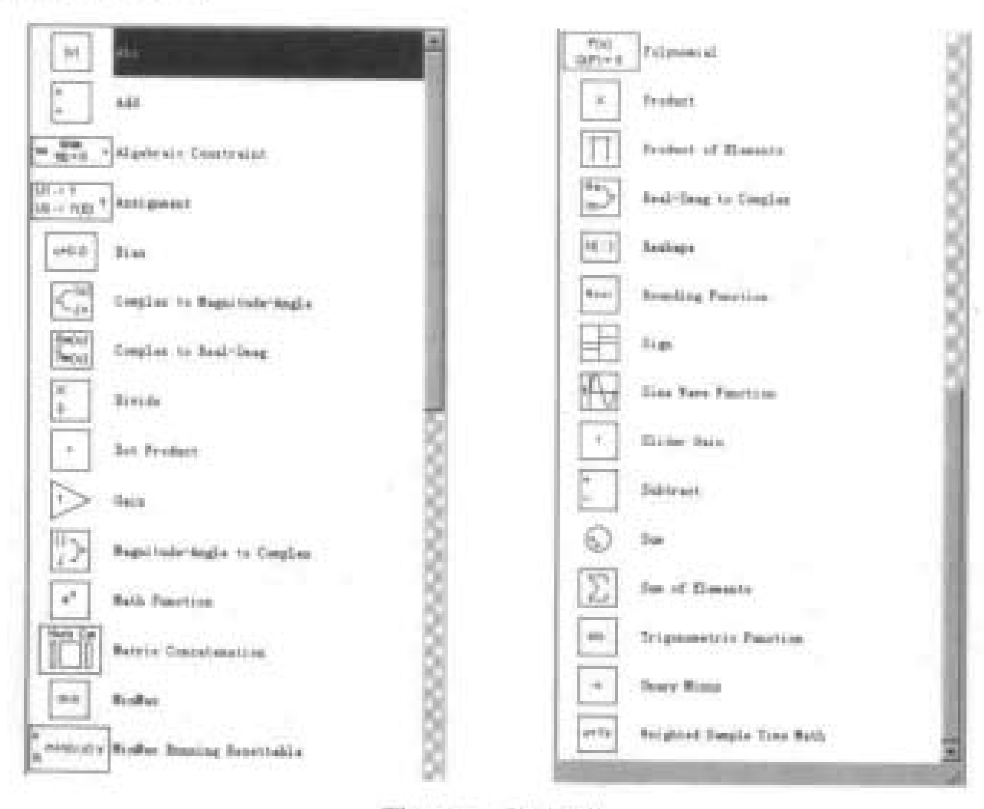


图 34-20　数学模块

（1）Abs 取绝对值；
（2）Add 加法；
（3）Algebraic Constraint 代数约束；
（4）Assignment 赋值；
（5）Bias 偏移；
（6）Complex to Magnitude-Angle 由复数输入转为幅值和相角输出；
（7）Complex to Real-Imag 由复数输入转为实部和虚部输出；
（8）Divide 除法；
（9）Dot Product 点乘运算；
（10）Gain 比例运算；
（11）Magnitude-Angle to Complex 由幅值和相角输入合成复数输出；



（12）Math Function 包括指数函数、对数函数、求平方、开根号等常用数学函数；
（13）Matrix Concatenation 矩阵级联；
（14）MinMax 最值运算；
（15）MinMax Running Resettable 最大最小值运算；
（16）Polynomial 多项式；
（17）Product 乘运算；
（18）Product of Elements 元素乘运算；
（19）Real-Imag to Complex 由实部和虚部输入合成复数输出；
（20）Reshape 取整；
（21）Rounding Function 舍入函数；
（22）Sign 符号函数；
（23）Sine Wave Function 正弦波函数；
（24）Slider Gain 滑动增益；
（25）Subtract 减法；
（26）Sum 求和运算；
（27）Sum of Elements 元素和运算；
（28）Trigonometric Function 三角函数，包括正弦、余弦、正切等；
（29）Unary Minus 一元减法；
（30）Weighted Sample Time Math 权值采样时间运算。

### 7．模型检测模块（Model Verification）

在 Simulink 基本模块中选择 Model Verification 后，点击便看到如图 34-21 所示的模型检测模块，它包括以下子模块：

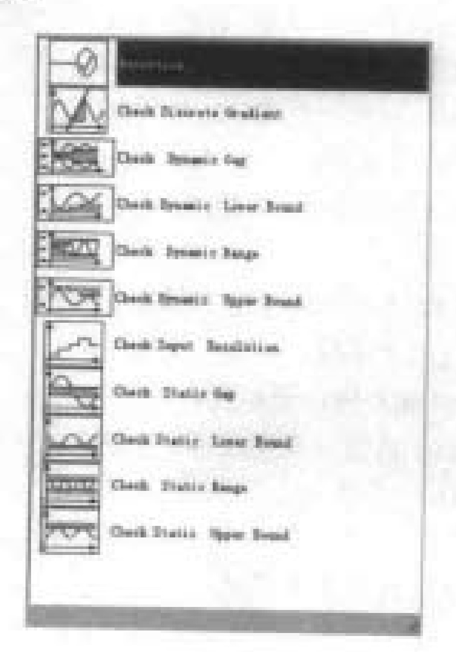


图 34-21　模型检测模块

（1）Assertion 确定操作；
（2）Check Discrete Gradient 检查离散梯度；
（3）Check Dynamic Gap 检查动态偏差；
（4）Check Dynamic Lower Bound 检查动态下限；
（5）Check Dynamic Range 检查动态范围；
（6）Check Dynamic Upper Bound 检查动态上限；
（7）Check Input Resolution 检查输入精度；
（8）Check Static Gap 检查静态偏差；
（9）Check Static Lower Bound 检查静态下限；
（10）Check Static Range 检查静态范围；
（11）Check Static Upper Bound 检查静态上限。

### 8．模型扩充模块（Model-Wide Utilities）

在 Simulink 基本模块中选择 Model-Wide Utilities 后，点击便看到如图 34-22 所示的模型扩充模块，它包括以下子模块：

（1）Block Support Table 功能块支持的表；
（2）DocBlock 文档模块；
（3）Model Info 模型信息；
（4）Timed-Based Linearization 时间线性分析；
（5）Trigger-Based Linearization 触发线性分析。

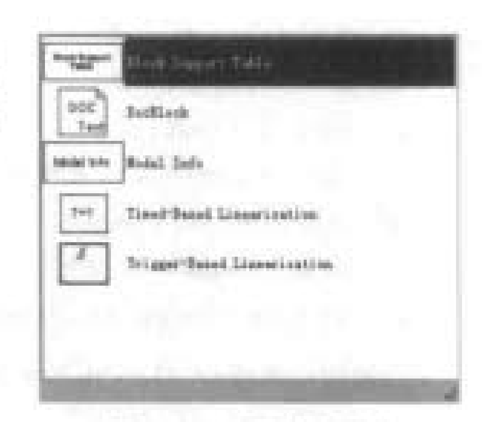



图 34-22　模型扩充模块

### 9．端口和子系统模块（Port & Subsystems）

在 Simulink 基本模块中选择 Port & Subsystems 后，点击便看到如图 34-23 所示的端口和子系统模块，它包括以下子模块：

（1）Configurable Subsystem 结构子系统；
（2）Atomic Subsystem 单元子系统；
（3）CodeReuseSubsystem 代码重用子系统；
（4）Enable 使能；
（5）Enabled and Triggered Subsystem 使能和触发子系统；
（6）Enabled Subsystem 使能子系统；
（7）For Iterator Subsystem 重复操作子系统；
（8）Function-Call Generator 函数响应生成器；
（9）Function-Call Subsystem 函数响应子系统；
（10）If 假设操作；
（11）If Action Subsystem 假设动作子系统；
（12）In1 输入端口；
（13）Model 模型；



（14）Out1 输出端口；

（15）Subsystem 子系统；

（16）Subsystem Examples 子系统例子；

（17）Switch Case 转换事件；

（18）Switch Case Action Subsystem 转换事件子系统；

（19）Trigger 触发操作；

（20）Triggered Subsystem 触发子系统；

（21）While Iterator Subsystem 重复子系统。

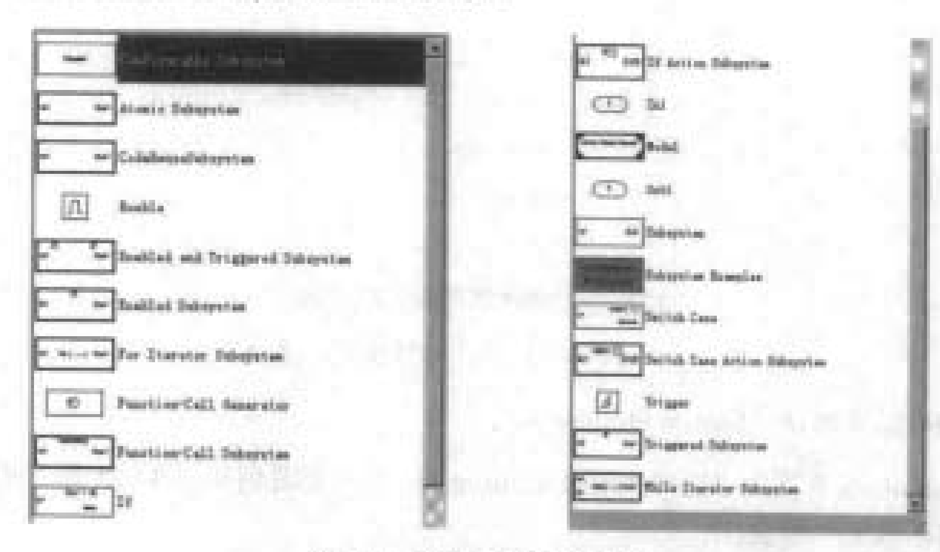


图 34-23　端口和子系统模块

## 10．信号属性模块（Signal Attributes）

在 Simulink 基本模块中选择 Signal Attributes 后，点击便看到如图 34-24 所示的信号属性模块，它包括以下子模块：

（1）Data Type Conversion 数据类型转换；

（2）Data Type Conversion Inherited 继承的数据类型转换；

（3）Data Type Duplicate 数据类型复制；

（4）Data Type Propagation 数据类型继承；

（5）Data Type Propagation Examples 数据类型继承例子；

（6）Data Type Scaling Strip 数据类型缩放；

（7）IC 信号输入属性；

（8）Probe 探针点；

（9）Rate Transition 比率变换；

（10）Signal Conversion 信号转换；

（11）Signal Specification 信号特征说明；

（12）Weighted Sample Time 权值采样时间；

（13）Width 信号宽度。

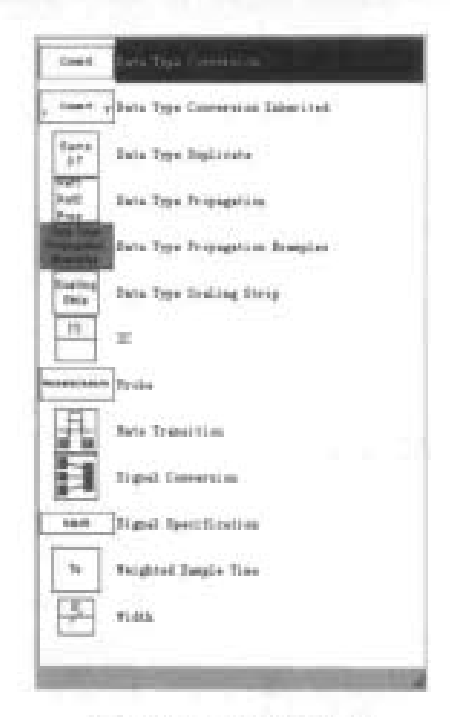


图 34-24　信号属性模块

11．信号路线模块（Signal Routing）

在 Simulink 基本模块中选择 Signal Routing 后，点击便看到如图 34-25 所示的信号路线模块，它包括以下子模块：

（1）Bus Assignment 总线分配；
（2）Bus Creator 总线生成；
（3）Bus Selector 总线选择；
（4）Data Store Memory 数据存储；
（5）Data Store Read 数据存储读取；
（6）Data Store Write 数据存储写入；
（7）Demux 将一个复合输入转化为多个单一输出；
（8）Environment Controller 环境控制器；
（9）From 信号来源；
（10）Goto 信号去向；
（11）Goto Tag Visibility 标签可视化；
（12）Index Vector 索引向量；
（13）Manual Switch 手动选择开关；
（14）Merge 信号合并；
（15）Multiport Switch 多端口开关；
（16）Mux 将多个单一输入转化为一个复合输出；
（17）Selector 信号选择器；



（18）Switch 开关选择，当第二个输入端大于临界值时，输出由第一个输入端而来，否则输出由第三个输入端而来。

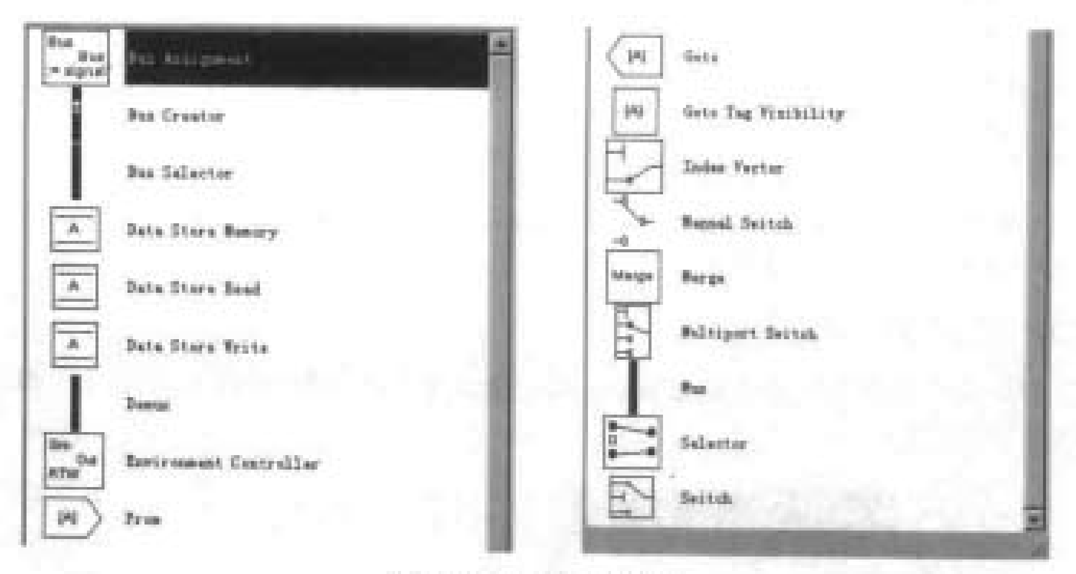



图 34-25 信号路线模块

## 12．接收器模块（Sinks）

在 Simulink 基本模块中选择 Sinks 后，点击便看到如图 34-26 所示的接收器模块，它包括以下子模块：

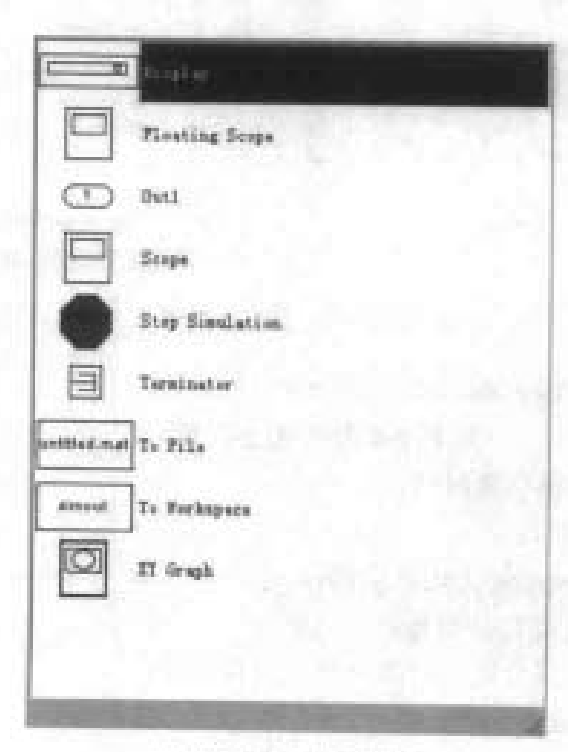



图 34-26 接收器模块

（1）Display 数字显示器；

（2）Floating Scope 浮动观察器；

（3）Out1 输出端口；

（4）Scope 示波器；

（5）Stop Simulation 仿真停止；

（6）Terminator 连接到没有连接到的输出端；

（7）To File 将输出数据写入数据文件保存；

（8）To Workspace 将输出数据写入 MATLAB 的工作空间；

（9）XY Graph 显示二维图形。

### 13. 输入源模块（Sources）

在 Simulink 基本模块中选择 Sources 后，点击便看到如图 34-27 所示的输入源模块，它包括以下子模块：



图 34-27　输入源模块

（1）Band-Limited White Noise 带限白噪声；

（2）Chirp Signal 产生一个频率不断增大的正弦波；

（3）Clock 显示和提供仿真时间；

（4）Constant 常数信号；

（5）Counter Free-Running 无限计数器；

（6）Counter Limited 有限计数器；

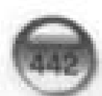
（7）Digital Clock 在规定的采样间隔产生仿真时间；

（8）From File 来自数据文件；

（9）From Workspace 来自 MATLAB 的工作空间；

（10）Ground 连接到没有连接到的输入端；

（11）In1 输入信号；







（12）Pulse Generator 脉冲发生器；

（13）Ramp 斜坡信号输入；

（14）Random Number 产生正态分布的随机数；

（15）Repeating Sequence 产生规律重复的任意信号；

（16）Repeating Sequence Interpolated 重复序列内插值；

（17）Repeating Sequence Stair 重复阶梯序列；

（18）Signal Builder 信号创建器；

（19）Signal Generator 信号发生器，可以产生正弦波、方波、锯齿波及任意波形；

（20）Sine Wave 正弦波信号；

（21）Step 阶跃信号；

（22）Uniform Random Number 一致随机数。

### 14. 用户自定义函数模块（User-Defined Functions）

在 Simulink 基本模块中选择 User-Defined Functions 后，点击便看到如图 34-28 所示的用户自定义函数模块，它包括以下子模块：

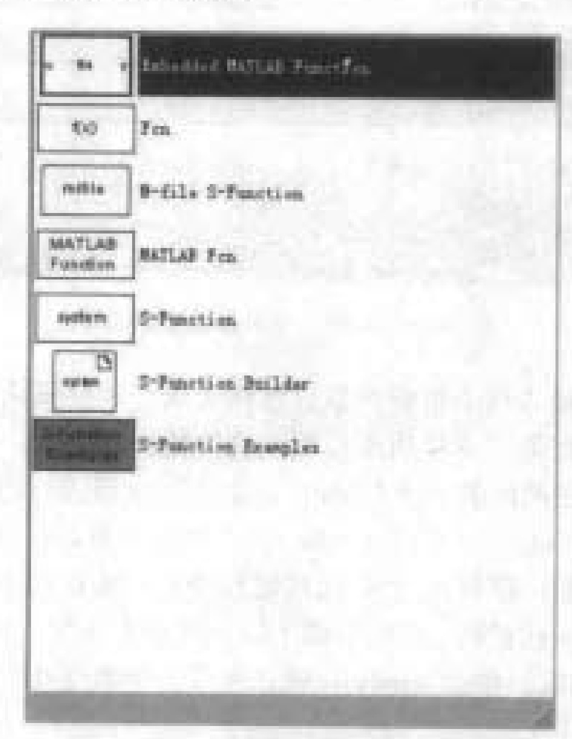



图 34-28　用户自定义函数模块

（1）Embedded MATLAB Function　嵌入的 MATLAB 函数；

（2）Fcn　用自定义的函数（表达式）进行运算；

（3）M-file S-Function　M 文件编写的 S 函数；

（4）MATLAB Fcn 利用 MATLAB 的现有函数进行运算；

（5）S-Function 调用自编的 S 函数程序进行运算；

（6）S-Function Builder　S 函数建立器；

（7）S-Function Examples　S 函数例子。

### 34.1.4　Simulink 功能模块的处理

#### 1．Simulink 模块参数设置

（1）功能模块参数设置

在设置功能模块参数后，才能进行仿真操作。不同功能模块的参数是不同的，用鼠标双击该功能模块自动弹出相应的参数设置对话框。图 34-29 是传输延迟功能模块的对话框。

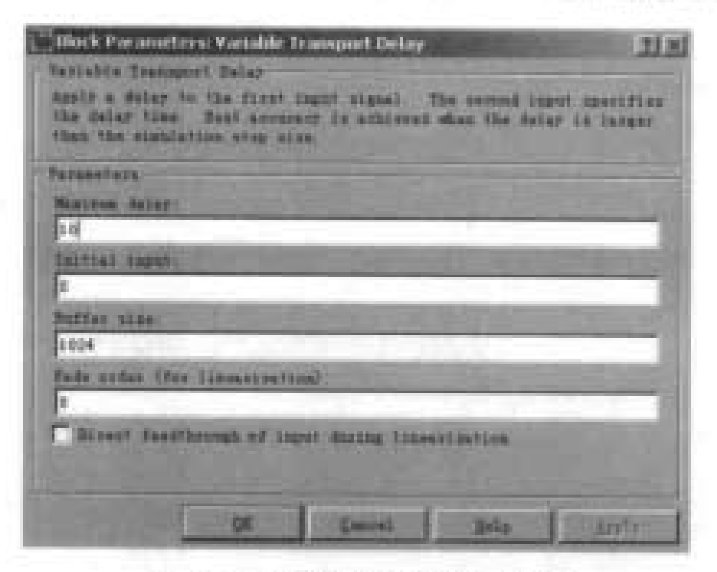


图 34-29　功能模块参数设置对话框

功能对话框由功能模块说明框和参数设置框组成。功能模块说明框用于说明该功能模块使用方法和功能；参数设置框用于设置该功能模块的参数。例如，传输延迟参数由最大延迟、初始输入、缓冲区的大小和 pade 近似的阶次组成，用户可输入相关参数。每个对话框的下面有 OK（确认）、Cancel（取消）、Help（帮助）和 Apply（应用）四个按钮，设置功能模块参数后，需单击 OK 按钮进行确认，将设置参数送到仿真操作画面，并关闭对话框。单击 Cancel 按钮，将取消刚才输入的设置参数，并关闭对话框。单击 Help 按钮，弹出 Web 求助画面。单击 Apply 按钮，将设置参数送仿真操作画面，但不关闭参数设置对话框。

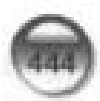
（2）示波器参数设置

采用 Simulink 仿真时，仿真结果通常通过示波器显示出来，因为示波器显示的结果直观方便。使用示波器显示结果时，需要对示波器的相关参数进行设置。示波器包括单踪和双踪示波器两种，下面分别对这两种示波器参数设置进行介绍。

① 单踪示波器参数设置：单踪示波器显示的是输入信号与时间关系的曲线。

设置方法如下：

双击已建立的示波器，将看到示波器显示界面，单击此对话框中工具条中的 ▣ 设置



图标，将弹出如图 34-30 所示的示波器属性对话框。示波器属性对话框中包括两个标签，可用于设置坐标窗口数、显示时间范围、标记和显示频度或采样时间等。

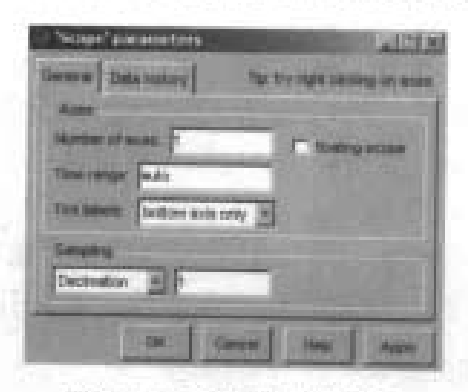


图 34-30　示波器属性对话框

② 双踪示波器参数设置：双踪示波器显示的是两个输入信号之间关系的曲线。

设置方法同单踪示波器，示波器属性对话框中包括两个标签，可用于设置 $x$ 坐标和 $y$ 坐标的显示时间范围、标记和显示频度或采样时间等。

### 2. Simulink 模块基本操作

功能模块的基本操作包括模块的移动、复制、删除、转向、改变大小、模块命名、颜色设定、参数设定、属性设定、模块输入输出信号等。模块库中的模块可以直接用鼠标进行拖拽（选中模块，按住鼠标左键不放）而放到模型窗口中进行处理。在模型窗口中，选中模块，则其 4 个角会出现黑色标记，此时可以对模块进行下述操作。

（1）移动：选中模块，按住鼠标左键将其拖拽到所需的位置即可。若要脱离线而移动，可按住 shift 键，再进行拖拽。

（2）复制：选中模块，然后按住鼠标右键进行拖拽即可复制同样的一个功能模块。

（3）删除：选中模块，按 Delete 键即可。若要删除多个模块，可以同时按住 Shift 键，再用鼠标选中多个模块，按 Delete 键即可；也可以用鼠标选取某区域，再按 Delete 键就可以把该区域中的所有模块和线等全部删除。

（4）转向：为了能够顺序连接功能模块的输入和输出端，功能模块有时需要转向。在菜单 Format 中，选择 Flip Block 旋转 180 度，选择 Rotate Block 顺时针旋转 90 度；或者直接按 Ctrl+F 键执行 Flip Block，按 Ctrl+R 键执行 Rotate Block。

（5）改变大小：选中模块，对模块出现的 4 个黑色标记进行拖拽即可。

（6）模块命名：先用鼠标在需要更改的名称上单击一下，然后直接更改即可。名称在功能模块上的位置也可以变换 180 度，可以用 Format 菜单中的 Flip Name 来实现，也可以直接通过鼠标进行拖拽。Hide Name 可以隐藏模块名称。

（7）颜色设定：Format 菜单中的 Foreground Color 可以改变模块的前景颜色，Background Color 可以改变模块的背景颜色；而模型窗口的颜色可以通过 Screen Color 来改变。

（8）参数设定：用鼠标双击模块，就可以进入模块的参数设定窗口，从而对模块进行参数设定。参数设定窗口包含了该模块的基本功能帮助，为获得更详尽的帮助，可以单击

其上的 help 按钮。通过对模块的参数设定，就可以获得需要的功能模块。

（9）属性设定：选中模块，打开 Edit 菜单的 Block Properties 可以对模块进行属性设定，包括对 Description，Priority，Tag，Open function，Attributes format string 等属性。其中 Open function 属性是一个很有用的属性，通过它指定一个函数名，则当该模块被双击之后，Simulink 就会调用该函数并执行，这种函数在 MATLAB 中称为回调函数。

（10）模块的输入输出信号：模块处理的信号包括标量信号和向量信号。标量信号是一种单一信号，而向量信号为一种复合信号，是多个信号的集合，它对应着系统中几条连线的合成。缺省情况下，大多数模块的输出都为标量信号，对于输入信号，模块都具有一种“智能”的识别功能，能自动进行匹配。某些模块通过对参数的设定，可以使模块输出向量信号。

#### 3．Simulink 模块间连线处理

Simulink 模型的构建是通过用线将各种功能模块进行连接而构成的。用鼠标可以在功能模块的输入与输出端之间直接连线。所画的线可以改变粗细、设定标签，也可以把线折弯、分支。

（1）改变粗细：线有粗细，是因为线引出的信号可以是标量信号或向量信号，当选中 Format 菜单下的 Wide Vector Lines 时，线的粗细会根据线所引出的信号是标量还是向量而改变，如果信号为标量则为细线，若为向量则为粗线。选中 Vector Line Widths 则可以显示出向量引出线的宽度，即向量信号由多少个单一信号合成。

（2）设定标签：只要在线上双击鼠标，即可输入该线的说明标签，也可以通过选中线，然后打开 Edit 菜单下的 Signal Properties 进行设定。其中 signal name 属性的作用是标明信号的名称，设置这个名称反映在模型上的直接效果，就是与该信号有关的端口相连的所有直线附近都会出现写有信号名称的标签。

（3）线的折弯：按住 Shift 键，再用鼠标在要折弯的线处单击一下，就会出现圆圈，表示折点，利用折点就可以改变线的形状。

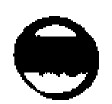
（4）线的分支：按住鼠标右键，在需要分支的地方拉出即可以；或者按住 Ctrl 键，并在要建立分支的地方用鼠标拉出。

## 34.2 Simulink 自定义功能模块

自定义功能模块有两种方法，一种方法是采用 Signal&Systems 模块库中的 Subsystem 功能模块，利用其编辑区设计组合新的功能模块；另一种方法是将现有的多个功能模块组合起来，形成新的功能模块。对于很大的 Simulink 模型，通过自定义功能模块可以简化图形，减少功能模块的个数，有利于模型的分层构建。

### 34.2.1 采用 Subsystem 建立自定义功能模块

将 Signal&Systems 模块库中的 Subsystem 功能模块复制到打开的模型窗口中。





双击 Subsystem 功能模块，进入自定义功能模块窗口，即可利用已有的基本功能模块设计出新的功能模块。

### 34.2.2 多个模块组合自定义功能模块

在模型窗口中建立所定义功能模块的子模块。用鼠标将这些需要组合的功能模块框住，然后选择 Edit 菜单下的 Create Subsystem 即可。

### 34.2.3 自定义功能模块的封装

上面提到的两种方法都只是创建一个功能模块而已，如果要命名该自定义功能模块，对功能模块进行说明，选定模块外观，设定输入数据窗口，则需要对其进行封装处理。

首先选中 Subsystem 功能模块，再打开 Edit 菜单中的 Mask Subsystem 进入 mask 的编辑窗口，可以看出有 Icon, Initialization, Documentation 这 3 个标签页。

#### 1. Icon 标签页

它用于设定功能模块外观，最重要的部分是 Drawing Commands，在该区域内可以用 disp 指令设定功能模块的文字名称，用 plot 指令画线，用 dpoly 指令画转换函数。

注意：尽管这些命令在名字上和以前讲的 MATLAB 函数相同，但它们在功能上却不完全相同，因此不能随便套用以前所讲的格式。

（1）disp('text')：在功能模块上显示设定的文字内容。

（2）disp('text1\ntext2')：分行显示文字 text1 和 text2。

（3）plot([x1 x2 ··· xn],[y1 y2 ··· yn])：在功能模块上画出由[x1 y1]经[x2 y2]经[x3 y3] ···直到[xn,yn]为止的直线。功能模块的左下角会根据目前的坐标刻度被正规化为[0,0]，右上角则会依据目前的坐标刻度被正规化为[1,1]。

（4）dpoly(num,den)：按 $s$ 次数的降幂排序，在功能模块上显示连续的传递函数。

（5）dpoly(num,den, 'z')：按 $z$ 次数的降幂排序，在功能模块上显示离散的传递函数。

用户还可以设置一些参数来控制图标的属性，这些属性在 Icon 页右下端的下拉式列表中进行选择。

（1）Icon frame：选择 Visible 则显示外框线；选择 Invisible 则隐藏外框线。

（2）Icon Transparency：选择 Opaque 则隐藏输入输出的标签；选择 Transparent 则显示输入输出的标签。

（3）Icon Rotation：旋转模块。

（4）Drawing coordinate：画图时的坐标系。

#### 2. Initialization 标签页

它用于设定输入数据窗口（Prompt List），主要是设计输入提示（prompt）以及对应的变量名称（variable）。在 prompt 栏上输入变量的含义，其内容会显示在输入提示中。而





variable 是仿真要用到的变量，该变量的值一直存于 mask workspace 中，因此可以与其他程序相互传递。

如果配合在 initialization commands 内编辑程序，可以发挥功能模块的功能来执行特定的操作。

（1）在 prompt 编辑框中输入文字，这些文字就会出现在 prompt 列表中；在 variable 列表中输入变量名称，则 prompt 中的文字对应该变量的说明。如果要增加新的项目，可以单击边上的 Add 按钮。Up 和 Down 按钮用于执行项目间的位置调整。

（2）Control type 列表给用户提供选择设计的编辑区，选择 Edit 会出现供输入的空白区域，所输入的值代表对应的 variable；Popup 则为用户提供可选择的列表框，所选的值代表 variable，此时在下面会出现 Popup strings 输入框，用来设计选择的内容，各值之间用逻辑或符号“|”隔开；若选择 Checkbox 则用于 on 与 off 的选择设定。

（3）Assignment 属性用于配合 Control type 的不同选择来提供不同的变量值，变量值分为 Evaluate 和 Literal 两种，其含义如表 34-1 所示。

表 34-1　Assignment 属性的含义

| Control Type | Assignment | |
|---|---|---|
| | Evaluate | Literal |
| Edit | 输入的文字是程序执行时所用的变量值 | 输入内容作字符串处理 |
| Popup | 所选序号，选第一项输出 1，往下类推 | 选择内容作字符串处理 |
| Checkbox | 输入为 1 或 0 | 输入为'on'或'off'的字符串 |

### 3. Documentation 标签页

它用于设计该功能模块的文字说明，主要用来针对完成的功能模块，编写相应的说明文字和 Help。

（1）在 Block description 中输入的文字，会出现在参数窗口的说明部分。

（2）在 Block help 中输入的文字，会显示在单击参数窗口中的 help 按钮后浏览器所加载的 HTML 文件中。

（3）在 Mask type 中输入的文字作为封装模块的标注性说明，在模型窗口下，将鼠标指向模块，则会显示该文字。当然，必须先在 View 菜单中选择 Block Data Tips——Show Block Data Tips。

## 34.3　S 函数设计与应用

前面讲到，Simulink 为用户提供了许多内置的基本库模块，如连续系统模块库（Continous）、离散系统模块库（Discontinuous），通过这些模块的连接构成系统的模型。这些内置的基本库模块是有限的，在许多情况下，尤其是在特殊的应用中，需要用到一些特殊的模块，这些模块可以用基本模块构成，由基本模块扩展而来。

Simulink 提供了一个功能强大的，对模块库进行扩展的新工具 S-Function，它依然是基于 Simulink 原来提供的内置模块，通过对那些经常使用的模块进行组合并封装，构建出可





重复使用的新模块。

S-Function 是系统函数（System Function）的简称，是一个动态系统的计算机语言描述。在 MATLAB 中，用户可以选择用 M 文件编写，也可以用 C 或 MEX 文件编写，在这里只给大家介绍如何用 M 文件编写 S-Function，使用 C 语言或 MEX 文件编写的方法与 M 文件编写的方法基本类似。

S-Function 提供了扩展 Simulink 模块库的有力工具，它采用一种特定的调用语法，实现函数和 Simulink 解法器之间的交互。

S-Function 最广泛的用途是定制用户自己的 Simulink 模块。它的形式十分通用，能够支持连续系统、离散系统和混合系统。

### 34.3.1 S 函数设计

对于一些算法比较复杂的模块，可以使用 MATLAB 语言按照 S-Function 的格式来编写代码。应该注意的是，这样构造的 S-Function 只能用于基于 Simulink 的仿真，并不能将其转换成独立于 MATLAB 的程序。

MATLAB 提供了一个模板文件，方便了 S-Function 的编写，该模板文件位于 MATLAB 根目录 toolbox/Simulink/blocks 下，去除注释部分后的程序如下所示。

S-Function 的模板

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
switch flag,
case 0,
% Initialization
[sys,x0,str,ts]=mdlInitializeSizes;
case 1,
% Derivatives
sys=mdlDerivatives(t,x,u);
case 2,
% Update
sys=mdlUpdate(t,x,u);
case 3,
% Outputs
sys=mdlOutputs(t,x,u);
case 4,
% GetTimeOfNextVarHit
sys=mdlGetTimeOfNextVarHit(t,x,u);
case 9,
% Terminate
sys=mdlTerminate(t,x,u);
otherwise
% Unexpected flags
error(['Unhandled flag = ',num2str(flag)]);
end
```

其中的子函数如下所示。

S-Function 的子函数

```
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 0;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];
function sys=mdlDerivatives(t,x,u)
sys = [];
function sys=mdlUpdate(t,x,u).
sys = [];
function sys=mdlOutputs(t,x,u)
sys = [];
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
function sys=mdlTerminate(t,x,u)
sys = [];
```

模板文件中 S-Function 的结构十分简单，它只为不同的 flag 值指定需调用的 M 文件子函数。如当 flag=3 时，即模块处于计算输出这个仿真阶段时，需调用的子函数为 sys=mdloutputs(t,x,u)。模板文件使用 switch 语句来完成这种指定，当然这种结构并不惟一，用户也可以使用 if 语句来完成同样的功能。在实际运用时，可以根据需要去掉某些值，因为并不是每个模块都需要经过所有的子函数调用。

模板文件只是 Simulink 为方便用户而提供的一种参考格式，并不是编写 S-Function 的语法要求，用户完全可以改变子函数的名称，或者直接把代码写在主函数中。但使用模板文件的好处是，比较方便，条理清晰。

使用模板编写 S-Function，用户只需把 S-函数名换成期望的函数名称，如果需要额外的输入参量，还需在输入参数列表的后面增加这些参数，因为前面的 4 个参数是 Simulink 调用 S-Function 时自动传入的。对于输出参数，最好不做修改。接下来的工作就根据所编 S-Function 要完成的任务，用相应的代码去替代模板里各个子函数的代码。

Simulink 在每个仿真阶段都会对 S-Function 进行调用，在调用时，Simulink 会根据所

处的仿真阶段为 flag 传入不同的值，而且还会为 sys 这个返回参数指定不同的角色，也就是说尽管是相同的 sys 变量，但在不同的仿真阶段其意义却不相同，这种变化由 Simulink 自动完成。

M 文件 S-Function 可用的子函数说明如下。

（1）mdlInitializeSizes：定义 S-Function 模块的基本特性，包括采样时间、连续或者离散状态的初始条件和 sizes 数组。

（2）mdlDerivatives：计算连续状态变量的微分方程。

（3）mdlUpdate：更新离散状态、采样时间和主时间步的要求。

（4）mdlOutputs：计算 S-Function 的输出。

（5）mdlGetTimeOfNextVarHit：计算下一个采样点的绝对时间，即在 mdlInitializeSizes 中说明了一个可变的离散采样时间。

（6）mdlTerminate：结束仿真任务。

概括说来，建立 S-Function 可以分成两个分离的任务：

（1）初始化模块特性，包括输入输出信号的宽度，离散连续状态的初始条件和采样时间。

（2）将算法放到合适的 S-Function 子函数中去。

为了让 Simulink 识别出一个 m 文件 S-Function，用户必须在 S-函数里提供有关 S-函数的说明信息，包括采样时间、连续或者离散状态个数等初始条件。这一部分主要是在 mdlInitializeSizes 子函数里完成。

Sizes 数组是 S-Function 函数信息的载体，其内部的字段意义分别为：

（1）NumContStates 连续状态的个数（状态向量连续部分的宽度）。

（2）NumDiscStates 离散状态的个数（状态向量离散部分的宽度）。

（3）NumOutputs 输出变量的个数（输出向量的宽度）。

（4）NumInputs 输入变量的个数（输入向量的宽度）。

（5）DirFeedthrough 有无直接馈入。注意：DirFeedthrough 是一个布尔变量，它的取值只有 0 和 1 两种。0 表示没有直接馈入，此时用户在编写 mdlOutputs 子函数时就要确保子函数的代码里不出现输入变量 $u$；1 表示有直接馈入。

（6）NumSampleTimes 采样时间的个数，也就是 ts 变量的行数，与用户对 ts 的定义有关。

如果字段代表的向量宽度为动态可变，则可以将它们赋值为－1。需要指出的是，由于 S-Function 会忽略端口，所以当有多个输入变量或多个输出变量时，必须用 mux 模块或 demux 模块将多个单一输入合成为一个复合输入向量或将一个复合输出向量分解为多个单一输出。

S-Function 默认的 4 个输入参数为 $t$、$x$、$u$ 和 flag，它们的次序不能变动，代表的意义分别为：

（1）$t$ 表示当前仿真时间，这个输入参数通常用于决定下一个采样时刻，或者在多采样速率系统中，用来区分不同的采样时刻点，并据此进行不同的处理。

（2）$x$ 表示状态向量，这个参数是必须的，甚至在系统中不存在状态时也是如此，它



的使用非常灵活。

（3）$u$ 表示输入向量。

（4）flag 是一个用于控制在每一个仿真阶段调用哪一个子函数的参数，由 Simulink 在调用时自动取值。

S-Function 默认的 4 个返回参数为 sys、x0、str 和 ts，它们的次序不能变动，代表的意义分别为：

（1）sys 是一个通用的返回参数，其所返回值的意义取决于 flag 的值；

（2）x0 是初始的状态值（没有状态时是一个空矩阵[]），这个返回参数只在 flag 值为 0 时才有效，其他时候都会被忽略；

（3）str 这个参数没有什么意义，是 MathWorks 公司为将来的应用保留的，M 文件 S-Function 必须把它设为空矩阵；

（4）ts 是一个 $m\times2$ 矩阵，它的两列分别表示采样时间间隔和偏移。

### 34.3.2 S 函数应用

下面用一个例子说明 S 函数的应用方法。

**例 34-1** 利用 MATLAB 中 S 函数模板设计一个离散系统的 S-Function。

**解**：程序代码如下所示：

MATLAB 程序代码

```
function [sys,x0,str,ts] = dsfunc(t,x,u,flag)
% 定义一个离散系统的 S-function
%     x(n+1) = Ax(n) + Bu(n)
%     y(n)   = Cx(n) + Du(n)
A=[-1.3839, -0.5097; 1.0000, 0];
B=[-2.5559, 0; 0, 4.2382];
C=[0, 2.0761; 0, 7.7891];
D=[-0.8141, -2.9334; 1.2426, 0];
switch flag,
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D);
  case 2,
    sys = mdlUpdate(t,x,u,A,B,C,D);
  case 3,
    sys = mdlOutputs(t,x,u,A,C,D);
  case 9,
    sys = []; % do nothing
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end
```



MATLAB 程序代码（续）

```
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)
sizes = simsizes;
sizes.NumContStates  = 0;
%两个离散状态
sizes.NumDiscStates  = size(A,1);
%两个输出
sizes.NumOutputs     = size(D,1);
%两个输入
sizes.NumInputs      = size(D,2);
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
%初始状态为 1
x0  = ones(sizes.NumDiscStates,1);
str = [];
%时间间隔为 1，时间偏移为 0
ts  = [1 0];
function sys = mdlUpdate(t,x,u,A,B,C,D)
sys = A*x+B*u;
sys = C*x+D*u;
```

## 34.4　Simulink 仿真举例

采用 Simulink 进行仿真，不仅系统模型的搭建简单方便，而且能直接获得系统输出或状态变量变化曲线，具有简单明了，直观形象的特点，得到了广泛应用。

**例 34-2**　已知一闭环系统结构如图 34-31 所示，其中，系统前向通道的传递函数为 $G(s)=\dfrac{s+0.5}{s+0.1}\bullet\dfrac{20}{s^3+12s^2+20s}$，而且前向通道有一个[-0.2,0.5]的限幅环节，图中用 $N$ 表示，反馈通道的增益为 1.5，系统为负反馈，阶跃输入经 1.5 倍的增益作用到系统，试利用 Simulink 对该闭环系统进行仿真，要求观测其单位阶跃响应曲线。

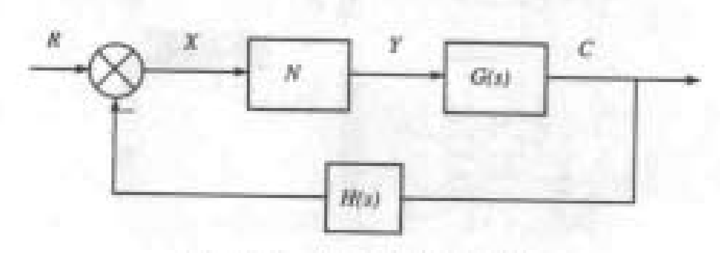



图 34-31　例 34-2 的系统框图

**解：**使用 Simulink 进行仿真的基本步骤如下所示。

（1）在 MATLAB7.0 的 MATLAB 窗中双击 Simulink 图标就打开 Simulink Library Browser

窗，在此窗口进入 File\New\Model，就会打开一个 untitled 窗（可以用 Save as 保存此窗口并改名），如图 34-30 所示。



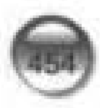
图 34-30　控制模型框图编辑窗口

（2）根据题意在 Simulink library Browser 中选定需要使用的子模块，如图 34-31 所示。在本例中，需要单位阶跃信号、增益模块、表示连续系统的模块、表示限幅环节的模块、用来把输入信号和输出信号组合起来以便直观观察的模块、把输入信号和反馈信号综合的模块、把仿真中的变量输出到工作空间的模块、观测系统响应曲线的模块和时钟模块。在 Simulink library Browser 窗口下找到了符合要求的模块，它们是：表示阶跃信号的模块 Step，位于 Source 模块组中；表示增益的模块 Gain，位于 Math Operations 模块组中；表示连续系统的模块 Transfer Fcn，位于 Continous 模块组中；表示限幅环节的模块 Saturation，位于 Discontinuities 模块组中；用来把输入信号和输出信号组合的模块 Mux，位于 Signal Routing 模块组中；用于把输入信号和反馈信号综合的模块 Sum，位于 Math Operations 模块组中；用于把仿真变量输出到工作空间的模块 To Workspace，位于 Sinks 模块组中；用于观测系统响应曲线的模块 Scope，位于 Sinks 模块组中；用来产生时钟信号的模块 Clock，位于 Sources 模块组中。



图 34-31　Simulink library Browser 窗口

把选定好的模块依次“拖”到 untitled 窗口中，如图 34-32 所示。注意，模块的选择有时不是惟一的，要根据自己的习惯定，如在本例中，表示系统前向通道传递函数的模块可





以用Transfer Fcn，也可以用Zero-Pole或State-Space，当然，需要进行简单的转换。

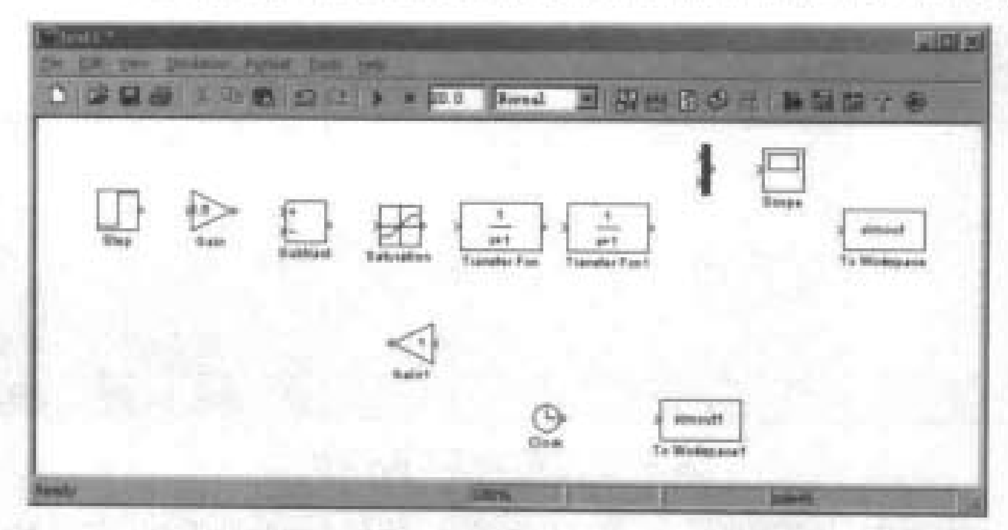


图34-32 模型编辑窗口

（3）连接模块并设定模块参数。把各功能模块按照逻辑关系连接起来，双击某一个模块，就会出现该模块的设置窗口，如图34-33所示，依次设置模块的参数。以图34-33为例，它是传递函数的设置窗口，设定它为$\frac{s+0.5}{s+0.1}$，则在Numerator中输入[1 0.5]，在Denominator和[1 0.1]]，其他的选项按默认值设定，然后单击“OK”按钮完成设置。

在Simulink仿真中常用的一个模块是To Workspace，它把Simulink仿真的数据传送到工作空间中，以供MATLAB程序使用。用鼠标双击“To Workspace”图标，得到图34-34的To Workspace模块参数对话框。

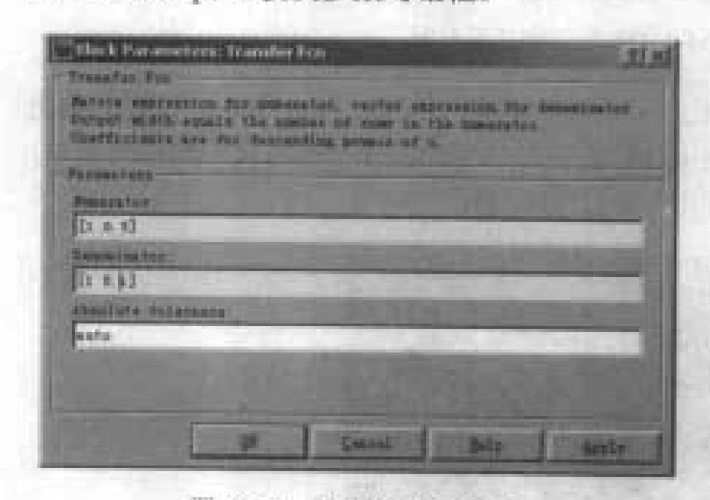


图34-33 模块参数设定窗口

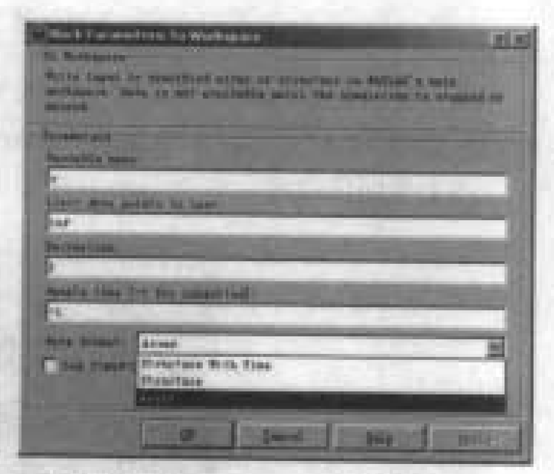
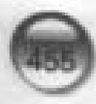


图34-34 To Workspace模块参数对话框

本题中，需要传输数据向量为$c$和$t$，以设置数据向量$c$为例，在Variable name编辑框中输入向量名$c$，save format编辑选择Array（向量）项，然后用鼠标单击“OK”按钮完成设置。仿真运行后，向量$c(t)$和$t$以各自变量名存在在MATLAB Workspace中。

（4）设置仿真器参数。仿真器设置窗口如图34-35所示，在Simulation\ Simulation





Parameters\Solver 中设置 SolverType、Solver（步长）、Stop Time 等。

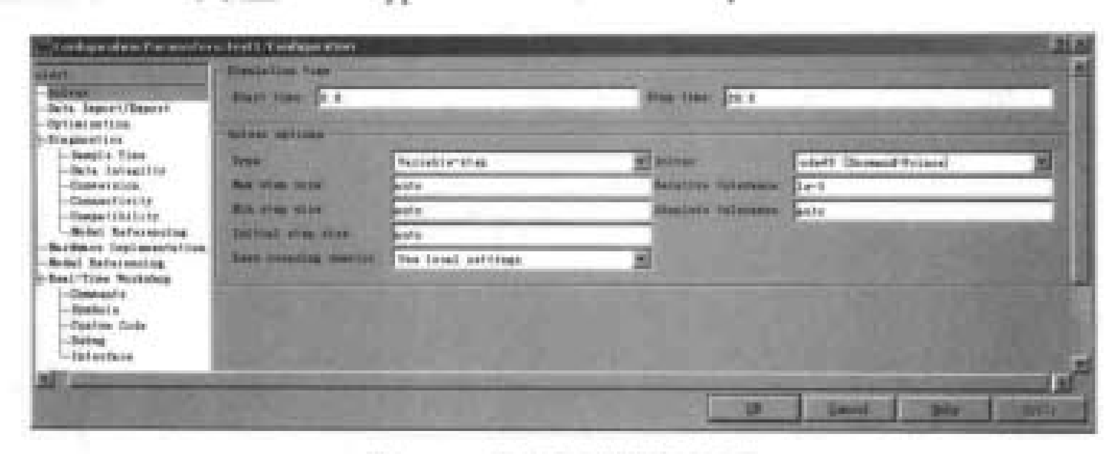


图 34-35 仿真器参数设置对话框

（5）运行仿真。模型编辑好后，单击 Start 按钮，运行 Simulation\Start，如图 34-36 所示。

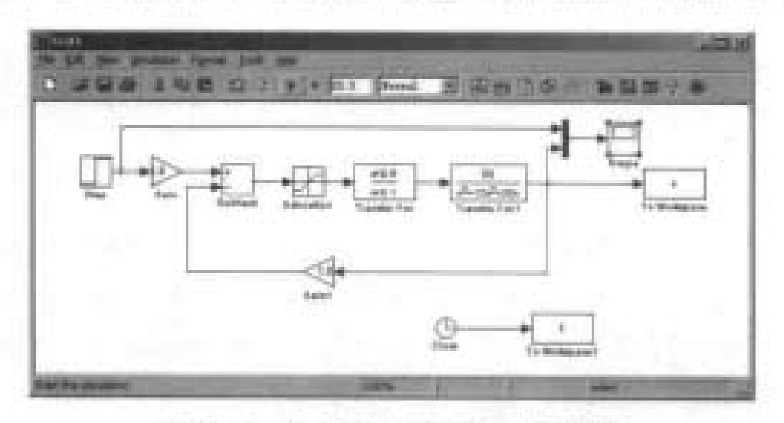


图 34-36 例 34-2 的 Simulink 仿真框图

（6）分析仿真结果。仿真中，一般采用示波器观测输出结果。双击 Scope 模块，输出的图形如图 34-37 所示。由于采用了 Mux 模块，把输入信号和输出信号合在一起同时显示，所以图上是两条曲线，它会自动分配不同的颜色以便观察。

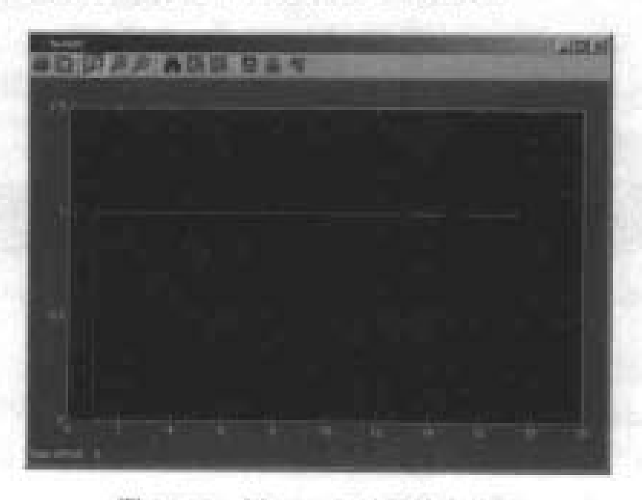


图 34-37 例 34-2 示波器输出结果



（7）对工作空间中的数据做后续处理。当仿真任务比较复杂时，需要把 Simulink 生成的数在导入到工作空间中来，做一些处理和分析。在本例中，Simulink 仿真结束后，输出结果通过 To workspace 传送到工作空间中，如图 34-38 所示，在工作空间中能看到这些变量，使用“whos”命令能看到这些变量的详细信息。另外，还有一些模块（如 From file，To file）能实现文件与 Simulink 之间的数据传输，读者可以通过模块的使用说明和有关对话框的引导进行使用。

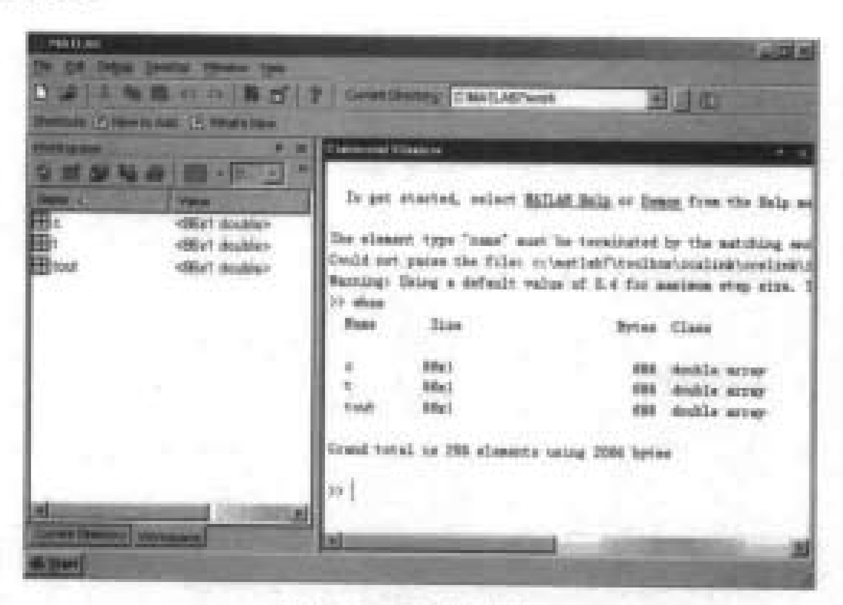



图 34-38　工作空间窗口

## 34.5　小结

本章介绍了 Simulink 的基本模块、基本功能，以及如何使用 Simulink 进行仿真等内容。Simulink 包含众多功能强大的模块，了解这些模块的主要内容，以及熟悉模块的基本操作，是使用 Simulink 进行仿真和设计的基础。



# 实　战　篇

- ◈ MATLAB 高等数学计算实例
- ◈ MATLAB 图形绘制实例
- ◈ MATLAB 扩展编程实例

# 第 35 章

# MATLAB 高等数学计算实例

MATLAB 作为强大的计算工具，在数学计算中有着广泛深入的应用。熟练掌握 MATLAB 这一工具在数学计算的应用，可以从繁杂的计算中解脱出来，大大提高效率。

## 35.1 极限运算

极限运算是高等数学的基础，MATLAB 提供了计算函数极限的命令，可方便用户进行极限运算。

例 35-1 求极限 $\lim\limits_{x\to 0}\dfrac{\tan(ax^2)}{2x^2+3(\sin x)^3}$。

解：在命令窗口中输入：

```
>> syms a x
y=tan(a*x^2)/(2*x^2+3*(sin(x))^3);
limit(y)
% 输出为
ans =
 1/2*a
```

可知 $\lim\limits_{x\to 0}\dfrac{\tan(ax^2)}{2x^2+3(\sin x)^3}=\dfrac{a}{2}$。

例 35-2 求极限 $\lim\limits_{x\to 1^+}\left[\dfrac{1}{x\ln^2 x}-\dfrac{1}{(x-1)^2}\right]$。

```
>> syms x
y=1/(x*(log(x)^2))-1/(x-1)^2;
limit(y,x,1,'right')
```

```
% 输出为
ans =
1/12
```

可知 $\lim_{x \to 1^+}\left[\frac{1}{x\ln^2 x}-\frac{1}{(x-1)^2}\right]=\frac{1}{12}$。

**例 35-3** 求极限 $\lim_{x \to \infty}(1+\frac{2}{n})^n$。

```
>> syms n
y=(1+2/n)^n;
limit(y,n,inf)
ans =
% 输出为
exp(2)
```

可知 $\lim_{x \to \infty}(1+\frac{2}{n})^n = e^2$。

## 35.2 求导数

### 35.2.1 一元函数求导

**例 35-4** 求 $y=\ln(x)$ 的一阶导数。

```
>> syms x
f=log(x);
diff(f);
% 输出为
ans =
 1/x
```

可知 $y'=\frac{1}{x}$。

**例 35-5** 求 $y=\ln(x)$ 的二阶导数。

```
>> syms x
f=log(x);
diff(f,2);
% 输出为
ans =
 -1/x^2
```

可知 $y''=-\frac{1}{x^2}$。

### 35.2.2 多元函数求导

**例 35-6** 已知函数 $z=3x^3y^2+\sin(xy)$，求 $\frac{\partial^2 z}{\partial x^3}$。

```
>> syms x y
z=3*x^3*y^2+sin(x*y);
diff(z,x,3);
% 输出为
ans =
 18*y^2-cos(x*y)*y^3
```

可知 $\frac{\partial^2 z}{\partial x^3}=18y^2-y^3\cos(xy)$。

**例 35-7** 已知函数 $z=3x^3y^2+\sin(xy)$，求 $\frac{\partial^2 z}{\partial x\partial y}$。

```
>> syms x y
z=3*x^3*y^2+sin(x*y);
dzx=diff(z,x);
dzxy=diff(dzx,y);
dzxy
pretty(diff(dzx,y))
 % 输出为
dzxy =
18*x^2*y-sin(x*y)*x*y+cos(x*y)
```

可知 $\frac{\partial^2 z}{\partial x\partial y}=18x^2y-xy\sin(xy)+\cos(xy)$。

**例 35-8** 求 $\frac{\mathrm{d}}{\mathrm{d}x}\begin{pmatrix}2a & t^3\\ t\sin x & \ln x\end{pmatrix}$。

```
>> syms a t x
A=[2*a,t^3;t*sin(x),log(x)];
dAx=diff(A,x);
>> dAx
  % 输出为
dAx =
 [        0,         0]
 [  t*cos(x),      1/x]
```

可知 $\frac{\mathrm{d}}{\mathrm{d}x}\begin{pmatrix}2a & t^3\\ t\sin x & \ln x\end{pmatrix}=\begin{pmatrix}0 & 0\\ t\cos x & \frac{1}{x}\end{pmatrix}$。



### 35.2.3　参数方程求导

**例 35-9**　已知一参数方程为$\begin{cases} x = t\sin t \\ y = t(1-\cos t) \end{cases}$，求$\dfrac{dy}{dx}$。

```
>> syms t
x=t*sin(t);
y=t*(1-cos(t));
dx=diff(x,t);
dy=diff(y,t);
dx
dy
dy/dx
%输出为
dx =
 sin(t)+t*cos(t)
  % 输出为
dy =
 1-cos(t)+t*sin(t)
%输出为
dy/dx =
 (1-cos(t)+t*sin(t))/(sin(t)+t*cos(t))
```

可知$\dfrac{dy}{dx} = \dfrac{1-\cos t + t\sin t}{\sin t + t\cos t}$。

### 35.2.4　隐函数求导

**例 35-10**　已知$e^y + y\sin x - e^x = 0$所确定的隐函数$y = y(x)$，求$\dfrac{dy}{dx}$。

```
>> syms x y
f=exp(y)+y*sin(x)-exp(x);
dfx=diff(f,x);
dfy=diff(f,y);
dyx=-dfx/dfy;
dyx;
% 输出为
dyx =
 (-y*cos(x)+exp(x))/(exp(y)+sin(x))
```

可知$\dfrac{dy}{dx} = \dfrac{-y\cos x + e^x}{e^y + \cos x}$。

### 35.2.5　求梯度与方向导数

**例 35-11**　求函数$f(x,y,z) = x^2 + 2y^2 + z^2$在点（−1，1，2）处的梯度。

```
>> syms x y z
f=x^2+2*y^2+z^2;
s=jacobian(f)
sx=subs(s,'x','-1');
sy=subs(sx,'y','1');
sz=subs(sy,'z','2');
g=vpa(sz)
 s =
 [ 2*x, 4*y, 2*z]
 % 输出为
 g =
 [ -2., 4., 4.]
```

可知 $\text{grad}f(-1,1,2)=-2i+4j+4k$ 。

**例 35-12** 求函数 $f(x,y,z)=xy^2-xyz+z^3$ 在点（1，1，2）处沿方向角为 $(\alpha=\frac{\pi}{3},\beta=\frac{\pi}{3},\gamma=\frac{\pi}{3})$ 方向的方向导数。

```
syms x y z
f=x*y^2-x*y*z+z^3;
s=jacobian(f)
sx=subs(s,'x','1');
sy=subs(sx,'y','1');
sz=subs(sy,'z','2');
g=vpa(sz)
a=pi/3;
b=pi/3;
c=pi/3;
L=g*[cos(a),cos(b),cos(c)]'
% 输出为
g =
 [ -1., 0., 11.]
 L =
 5.00000
```

可知 $\text{grad}f(1,1,2)=-i+11k$ ，方向导数为 5。

## 35.3 求积分

### 35.3.1 定积分

**例 35-13** 求定积分 $\int_{-1}^{1}(x^2+3)^{\frac{1}{2}}\mathrm{d}x$ 。

```
>> syms x
f=sqrt(x^2+3);
int(f,x,-1,1);
>> ans
% 输出为
ans =
2+3/2*log(3)
```





可知$\int_{-1}^{1}(x^2+3)^{\frac{1}{2}}dx=2+\frac{3}{2}\log(3)$。

## 35.3.2　广义积分

例 35-14　计算广义积分$\int_{1}^{\infty}\frac{1}{x^3}dx$。

```
>> syms x
f=1/(x^3);
int(f,x,1,inf);
>> ans
% 输出为
ans =
 1/2
```

可知$\int_{1}^{\infty}\frac{1}{x^3}dx=\frac{1}{2}$。

## 35.3.3　重积分

例 35-15　计算$f(x,y)=e^{-\frac{x^2}{3}}\sin(x^2+2y)$在区间[−1,1]×[−1,1]上的二重积分。

```
>> syms x y
f = @(x,y)exp(-x.^2/3).*sin(x.^2+2*y);
I=dblquad(f,-1,1,-1,1)
% 输出为
I =
    0.4658
```

可知该值为 0.4658。

## 35.3.4　不定积分

例 35-16　计算$\int\frac{1}{x^2}dx$。

```
>> syms x
f=1/x^2
int(f);
 %输出为
f =
 1/x^2
 >> ans
 %输出为
ans =
 -1/x
```

可知$\int\frac{1}{x^2}dx=-\frac{1}{x}+C$，$C$ 为常数。

## 35.4 级数

### 35.4.1 级数展开

例 35-17 将函数 $f(x)=\dfrac{2}{x^2}$ 展开为 $(x-1)$ 的最高次为 3 的幂级数。

```
>> syms x
f=2/x^2
taylor(f,3,x,1)
%输出为
 f =
 2/x^2
%输出为
 ans =
 6-4*x+6*(x-1)^2
```

可知函数 $f(x)=\dfrac{2}{x^2}$ 在（x−1）展开最高次为 3 的幂级数为 $6(x-1)^2-4x+b$。

### 35.4.2 级数求和

例 35-18 求和 $\sum_{n=0}^{50}\left[an^3+(a-1)n^2\right]$。

```
>> syms a n
f=a*n^3+(a-1)*n^2
symsum(f,n,0,50)
 %输出为
f =
 a*n^3+(a-1)*n^2
 %输出为
ans =
-42925+1668550*a
```

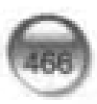
可知 $\sum_{n=0}^{50}\left[an^3+(a-1)n^2\right]=-42925+1668550*\mathrm{a}$ 。

## 35.5 求函数的零点和极值点

### 35.5.1 求函数的零点

例 35-19 求函数 $f(x)=x-2x+5$ 在 $x=2$ 附近的零点，并画出函数的图像。

```
>> f=@(x)x.^3-2*x-5
z=fzero(f,2)
```





```
x=0:0.1:4
f=@(x)x.^3-2*x-5
plot(x,f(x),'b',z,f(z),'rp')
axis([0,4,-100,100])
legend('f(x)','零点')
%输出为
z =
    2.0946
```

可知 $f(x)=x^3-2x+5$ 在 $x=2$ 附近的零点为2.0946，绘制的函数图形如图35-1所示。

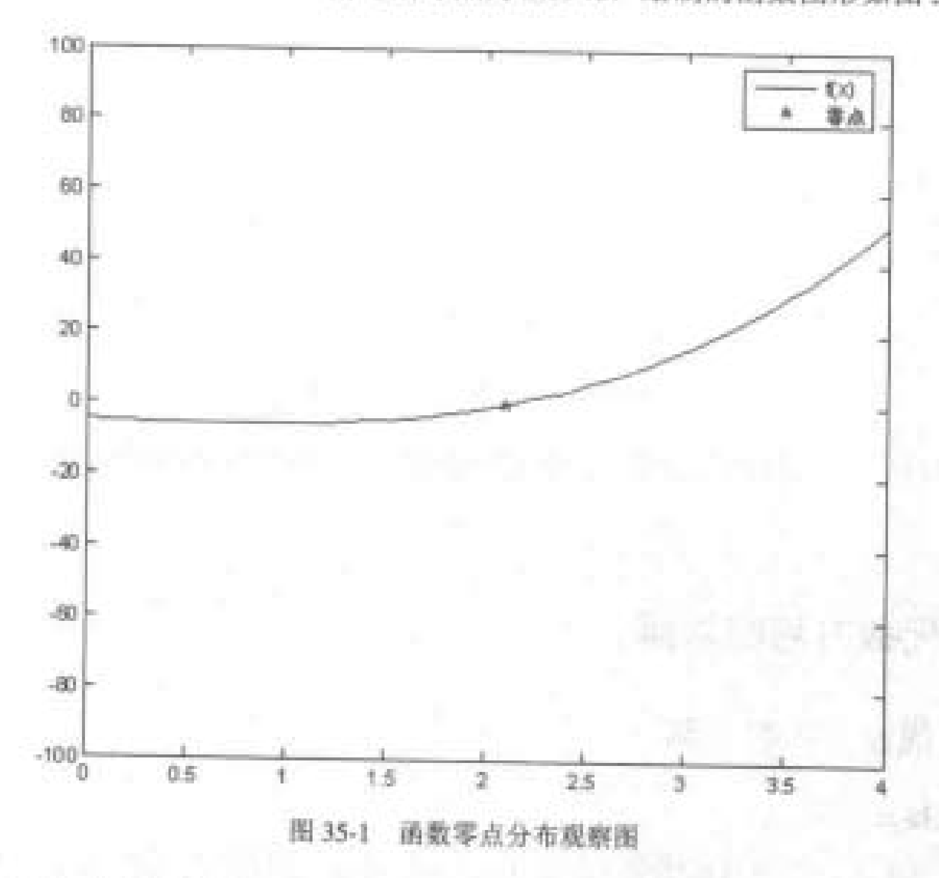



图35-1　函数零点分布观察图

### 35.5.2　求函数的极值点

**例35-20**　求函数 $f(x)=x-2x+5$ 在区间(0,2)内的极小值点。

```
>>  f=@(x)x.^3-2*x-5
z = fminbnd(f, 0, 2)
x=0:0.1:2
f=@(x)x.^3-2*x-5
plot(x,f(x),'b',z,f(z),'rp')
axis([0,2,-50,50])
legend('f(x)','极小点')
%输出为
x =
    0.8165
```

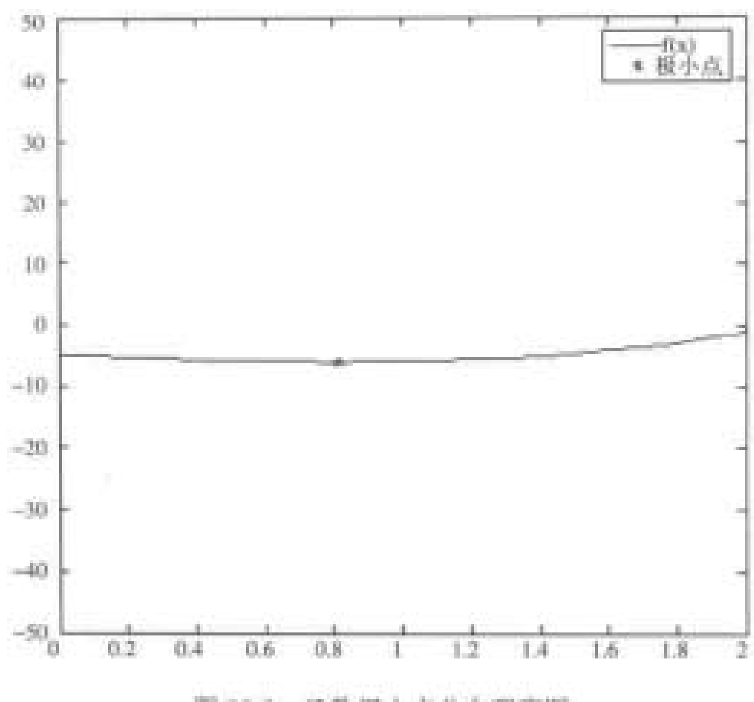



图 35-2　函数极小点分布观察图

可知 $f(x)=x^3-2x+5$ 在区间(0,2)内的极小值点为 0.8165，绘制的函数图形如图 35-2 所示。

# 35.6　代数方程组求解

## 35.6.1　线性方程组求解

### 1．直接法求解

MATLAB 中，采用直接法求解方程 $A_{m\times n}x_{n\times r}=B_{m\times r}$ 非常简单，用“左除”符号“\”便可实现求解，即 **x=A\B**。需要注意的是，**A** 在变量 **x** 的左边，对应的 **A** 必须在“\”的左边。

**例 35-21**　采用直接法求解下列方程组。

$$\begin{bmatrix}3 & 1 & -4 & 1\\ 1 & -5 & 0 & 7\\ 0 & 2 & 1 & -1\\ 1 & 6 & -1 & -4\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}=\begin{bmatrix}13\\ -9\\ 6\\ 0\end{bmatrix}$$

解：

```
A=[3,1,-4,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4]; %系数矩阵 A
b=[13,-9,6,0]'; %矩阵 B
x=A\b     %直接法求解
%输出为
x =
```



```
   43.6667
  -17.0000
   20.3333
  -19.6667
```

可知方程的解$[x_1, x_2, x_3, x_4]$=[ 43.6667, −17.0000, 20.3333, −19.6667]。

2．矩阵分解法求解

矩阵分解常用于方程求根，矩阵分解是指根据一定的原理，用某种算法将一个矩阵分解成若干个矩阵的乘积。常见的矩阵分解有LU分解、QR分解、Cholesky分解，Schur分解、Hessenberg分解和奇异分解等。

表35-1列出了MATLAB中一些常用的矩阵分解运算函数。

表35-1　常用矩阵分解运算函数

| 函 数 名 | 功能说明 |
|---|---|
| eig() | 矩阵的特征值分解 |
| qr() | 矩阵的QR分解 |
| schur() | 矩阵的Schur分解 |
| svd() | 矩阵的奇异值分解 |
| chol() | 矩阵的Cholesky分解 |
| lu() | 矩阵的LU分解 |

（1）LU分解

矩阵的LU分解就是将一个矩阵表示为一个变换下三角矩阵和一个上三角矩阵的乘积形式。线性代数中已经证明，只要方阵$\boldsymbol{A}$是非奇异的，LU分解总是可以进行的。

MATLAB提供的lu函数用于对矩阵进行LU分解，其调用格式为：

$$[\boldsymbol{L},\boldsymbol{U}]=\mathrm{lu}(\boldsymbol{X})$$

产生一个上三角阵$\boldsymbol{U}$和一个变换形式的下三角阵$\boldsymbol{L}$（行交换），使之满足$\boldsymbol{X}=\boldsymbol{LU}$。注意，这里的矩阵X必须是方阵。

$$[\boldsymbol{L},\boldsymbol{U},\boldsymbol{P}]=\mathrm{lu}(\boldsymbol{X})$$

产生一个上三角阵$\boldsymbol{U}$和一个下三角阵$\boldsymbol{L}$以及一个置换矩阵$\boldsymbol{P}$，使之满足$\boldsymbol{PX}=\boldsymbol{LU}$。当然矩阵$\boldsymbol{X}$同样必须是方阵。

实现LU分解后，线性方程组$\boldsymbol{Ax}=\boldsymbol{b}$的解$\boldsymbol{x}=\boldsymbol{U}\backslash(\boldsymbol{L}\backslash\boldsymbol{b})$或$\boldsymbol{x}=\boldsymbol{U}\backslash(\boldsymbol{L}\backslash\boldsymbol{Pb})$，这样可以大大提高运算速度。

**例35-22**　LU分解法求解下列方程组。

$$\begin{bmatrix} 3 & 1 & -4 & 1 \\ 1 & -5 & 0 & 7 \\ 0 & 2 & 1 & -1 \\ 1 & 6 & -1 & -4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}=\begin{bmatrix} 13 \\ -9 \\ 6 \\ 0 \end{bmatrix}$$

解：

```
A=[3,1,-4,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];  %系数矩阵A
```

```
b=[13,-9,6,0]';  %矩阵 B
[L,U]=lu(A);  %LU 分解
x=U\(L\b)    % 求解
[L,U ,P]=lu(A); %  采用 LU 的第 2 种格式分解
x=U\(L\P*b)    % 求解
%x 输出为
x =
   43.6667
  -17.0000
   20.3333
  -19.6667
%x1 输出为
x1 =
   43.6667
  -17.0000
   20.3333
  -19.6667
```

可知方程的解$[x_1, x_2, x_3, x_4]$ = [ 43.6667, −17.0000, 20.3333, −19.6667]。

（2）QR 分解

对矩阵 ***X*** 进行 QR 分解，就是把 ***X*** 分解为一个正交矩阵 ***Q*** 和一个上三角矩阵 ***R*** 的乘积形式。***QR*** 分解只能对方阵进行。

MATLAB 的函数 qr 可用于对矩阵进行 QR 分解，其调用格式为：

[***Q***,***R***]=qr(***X***)

产生一个正交矩阵 ***Q*** 和一个上三角矩阵 ***R***，使之满足 ***X***=***QR***。

[***Q***,***R***,***E***]=qr(***X***)

产生一个正交矩阵 ***Q***、一个上三角矩阵 ***R*** 以及一个置换矩阵 ***E***，使之满足 ***XE***=***QR***。

实现 QR 分解后，线性方程组 ***Ax***=***b*** 的解 ***x***=***R***\(***Q***'***b***)或 ***x***=***E***(***R***\(***Q***'***b***))。

**例 35-23** QR 分解法求解下列方程组。

$$\begin{bmatrix} 3 & 1 & -4 & 1 \\ 1 & -5 & 0 & 7 \\ 0 & 2 & 1 & -1 \\ 1 & 6 & -1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 13 \\ -9 \\ 6 \\ 0 \end{bmatrix}$$

解：

```
A=[3,1,-4,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4]; %系数矩阵 A
b=[13,-9,6,0]';  %矩阵 B
[Q,R]=qr(A);  %QR 分解
x=R\(Q\b)      % 求解
[Q,R,E]=qr(A); %  采用 QR 的第 2 种格式分解
x1=E*(R\(Q\b))    % 求解
%x 输出为
x =
   43.6667
  -17.0000
   20.3333
  -19.6667
```



```
%x1 输出为
x1 =
   43.6667
  -17.0000
   20.3333
  -19.6667
```

可知方程的解$[x_1, x_2, x_3, x_4]$=[ 43.6667, -17.0000, 20.3333, -19.6667]。

（3）Cholesky 分解

如果矩阵 ***X*** 是对称正定的，则 Cholesky 分解将矩阵 ***X*** 分解成一个下三角矩阵和上三角矩阵的乘积。设上三角矩阵为 ***R***，则下三角矩阵为其转置，即 ***X=R'R***，当限定 ***R*** 的对角元素为正时，这种分解是惟一的。

MATLAB 函数 chol(***X***)用于对矩阵 ***X*** 进行 Cholesky 分解，其调用格式为：

***R***=chol(***X***)

产生一个上三角阵 ***R***，使 ***R'R=X***。若 ***X*** 为非对称正定，则输出一个出错信息。

[***R***,*p*]=chol(***X***)

这个命令格式将不输出出错信息。

当 ***X*** 为对称正定的，则 *p*=0，**R** 与上述格式得到的结果相同；否则 *p* 为一个正整数。如果 ***X*** 为满秩矩阵，则 **R** 为一个阶数为 *q*=*p*-1 的上三角阵，且满足 ***R'R=X***(1:*q*,1:*q*)。实现 Cholesky 分解后，线性方程组 ***Ax=b*** 变成 ***R'Rx=b***，所以 ***x=R***\(***R'******b***)。

**例 35-24** Cholesky 分解法求解下列方程组。

$$\begin{bmatrix} 3 & 1 & -4 & 1 \\ 1 & -5 & 0 & 7 \\ 0 & 2 & 1 & -1 \\ 1 & 6 & -1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 13 \\ -9 \\ 6 \\ 0 \end{bmatrix}$$

解：

```
A=[3,1,-4,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4]; %系数矩阵A
b=[13,-9,6,0]';  %矩阵B
R=chol(A)  % 进行Cholesky 分解
%输出为
??? Error using ==> chol
Matrix must be positive definite.
```

可知，命令执行时，出现错误信息，说明 ***A*** 为非正定矩阵，不能用 Cholesky 分解法进行求解。

**例 35-25** Cholesky 分解法求解下列方程组。

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 13 \\ -9 \\ 6 \end{bmatrix}$$

解：

```
a=[4 -1 1;-1 4.25 2.75;1 2.75 3.5]; %系数矩阵A
b=[13,-9,6]';  %矩阵B
```

```
R=chol(a)    % 进行 Cholesky 分解
%输出为
R =
    2.0000   -0.5000    0.5000
         0    2.0000    1.5000
         0         0    1.0000
接着输入
x=R\(R'\b) % 求解
%x 输出为
x =
   -0.1992
   -6.7344
    7.0625
```

可知方程的解$[x_1, x_2, x_3]$=[ -0.1992，-6.7344，7.0625]。

3. 迭代法求解

直接法和矩阵分解法得到的解是理论上准确的，但是它们的计算量都是$n^3$数量级，存储量为$n^2$量级，这在$n$比较小的时候还比较合适（$n<400$），但是对于现在的很多实际问题，往往要求解很大的$n$的矩阵，而且这些矩阵往往是系数矩阵含有大量的0元素。对于这类矩阵，在用直接法时就会耗费大量的时间和存储单元。

因此，有必要引入一类新的方法：迭代法。迭代法的特点是速度快。与非线性方程的迭代方法一样，它需要构造一个等价的方程，从而构造一个收敛序列，序列的极限值就是方程组的根。

迭代解法非常适合求解大型系数矩阵的方程组。在数值分析中，迭代解法主要包括Jacobi 迭代法、Gauss-Serdel 迭代法、超松弛迭代法和两步迭代法。

（1）Jacobi 迭代法

对于线性方程组$\boldsymbol{Ax=b}$，如果$\boldsymbol{A}$为非奇异方阵，即$a_{ii}\neq 0(i=1,2,\cdots,n)$，则可将$\boldsymbol{A}$分解为：

$$\boldsymbol{A=D-L-U}$$

其中$\boldsymbol{D}$为对角阵，其元素为$\boldsymbol{A}$的对角元素，$\boldsymbol{L}$与$\boldsymbol{U}$为$\boldsymbol{A}$的下三角阵和上三角阵。

于是$\boldsymbol{Ax=b}$化为：

$$\boldsymbol{x=D^{-1}(L+U)x+D^{-1}b}$$

与之对应的迭代公式为：

$$\boldsymbol{x(k+1)=D^{-1}(L+U)x(k)+D^{-1}b}$$

这就是 Jacobi 迭代公式。如果序列$\{\boldsymbol{x}(k+1)\}$收敛于$\boldsymbol{x}$，则$\boldsymbol{x}$必是方程$\boldsymbol{Ax=b}$的解。

按照 Jacobi 迭代法的原理，在 MATLAB 中编写出此算法的函数文件 Jacobi.m 如下：

```
function [y,n]=jacobi(A,b,x0,eps)
if nargin==3
    eps=1.0e-6;
elseif nargin<3
    error
    return
end
D=diag(diag(A));    %求A的对角矩阵
L=-tril(A,-1);      %求A的下三角阵
```



```
U=-triu(A,1);          %求A的上三角阵
B=D\(L+U);
f=D\b;
y=B*x0+f;
n=1;                   %迭代次数
while norm(y-x0)>=eps
    x0=y;
    y=B*x0+f;
    n=n+1;
end
```

例35-26　用Jacobi迭代法求解下列线性方程组。设迭代初值为0，迭代精度为$10^{-6}$。

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 13 \\ -9 \\ 6 \end{bmatrix}$$

解：在命令中调用函数文件Jacobi.m，命令如下：

```
A=[4 -1 1;-1 4.25 2.75;1 2.75 3.5];   %系数矩阵A
b=[13,-9,6]';   %矩阵b
[x,n]=jacobi(A,b,[0,0,0]',1.0e-6)
%x输出为
x =
   -0.1992
   -6.7344
    7.0625
%迭代次数为
n =
    97
```

可知方程的解$[x_1, x_2, x_3]$=[ -0.1992，-6.7344，7.0625]。

（2）Gauss-Serdel迭代法

将Jacobi的迭代公式：

$$\boldsymbol{D}\boldsymbol{x}(k+1)=(\boldsymbol{L}+\boldsymbol{U})\boldsymbol{x}(k)+\boldsymbol{b}$$

改进为：

$$\boldsymbol{D}\boldsymbol{x}(k+1)=\boldsymbol{L}\boldsymbol{x}(k+1)+\boldsymbol{U}\boldsymbol{x}(k)+\boldsymbol{b}$$

于是得到：

$$\boldsymbol{x}(k+1)=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}\boldsymbol{x}(k)+(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{b}$$

该式即为Gauss-Serdel迭代公式。和Jacobi迭代相比，Gauss-Serdel迭代用新分量代替旧分量，精度会高些。

按照Gauss-Serdel迭代法的原理，在MATLAB中编写出此算法的函数文件gauseidel.m如下：

```
function [y,n]=gauseidel(A,b,x0,eps)
if nargin==3
    eps=1.0e-6;
elseif nargin<3
    error
    return
```

```
end
D=diag(diag(A));     %求A的对角矩阵
L=-tril(A,-1);       %求A的下三角阵
U=-triu(A,1);        %求A的上三角阵
G=(D-L)\U;
f=(D-L)\b;
y=G*x0+f;
n=1;                 %迭代次数
while norm(y-x0)>=eps
    x0=y;
    y=G*x0+f;
    n=n+1;
end
```

**例 35-27** 用 Gauss-Serdel 迭代法求解下列线性方程组。设迭代初值为 0，迭代精度为 $10^{-6}$。

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 13 \\ -9 \\ 6 \end{bmatrix}$$

解：在命令中调用函数文件 gauseidel.m 命令如下：

```
A=[4 -1 1;-1 4.25 2.75;1 2.75 3.5];  %系数矩阵A
b=[13,-9,6]';  %矩阵B
[x,n]=gauseidel(A,b,[0,0,0]',1.0e-6)
%x 输出为
x =
   -0.1992
   -6.7344
    7.0625
%迭代次数为
n =
    52
```

可知方程的解$[x_1, x_2, x_3]$=[ -0.1992，-6.7344，7.0625]。

## 35.6.2 非线性方程组求解

对于非线性方程组 $F(X)=0$，用 fsolve 函数求其数值解。fsolve 函数的调用格式为：

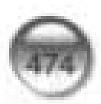
$$X=\text{fsolve('fun'},X0,\text{option})$$

其中 $X$ 为返回的解，fun 是用于定义需要求解的非线性方程组的函数文件名，$X0$ 是求根过程的初值，option 为最优化工具箱的选项设定。最优化工具箱提供了 20 多个选项，用户可以使用 optimset 命令将它们显示出来。

**例 35-28** 求下列非线性方程组 $\begin{cases} x-0.4\cos x-0.3\sin y=0 \\ y-0.6\cos x+0.5\sin y=0 \end{cases}$ 在(0.5,0.4)附近的数值解。

解：首先建立函数文件 myfun.m。

```
function q=myfun(p)
```





```
x=p(1);
y=p(2);
q(1)=x-0.4*cos(x)-0.3*sin(y);
q(2)=y-0.6*cos(x)+0.5*sin(y);
然后在给定的初值 x0=0.5,y0=0.4 下，调用 fsolve 函数求方程的根。
x=fsolve('myfun',[0.5,0.4]',optimset('Display','off'))
%x 输出为
x =
    0.4636
    0.3604
```

将求得的解代回原方程，可以检验结果是否正确，命令如下：

```
q=myfun(x)
%输出为
q =
  1.0e-007 *
    0.1544    0.2202
```

可见得到了较高精度的结果。

## 35.7 常微分方程求解

### 35.7.1 常微分方程的符号解

**例 35-29** 求 $\frac{dy}{dx}=3y^2$ 的解

```
>> dsolve('Dy=3*y^2','x')
 %输出为
ans =
-1/(3*x-C1)
```

可见方程的解为 $y=-\frac{-1}{3x-c1}$。

### 35.7.2 常微分方程组数值解

**例 35-30** 设有常微分方程 $\begin{cases} y'=f(t,y)=\frac{(y^2-t-2)(t+1)}{4} \\ y(0)=2 \end{cases}$，试求其数值解，并与精确解相比较。

解：首先建立函数文件 func.m。

```
function yp=func(t,y)
yp=(y^2-t-2)/4/(t+1);
```

然后求解微分方程。

```
t0=0;tf=5;
y0=2;
[t,y]=ode23('func',[t0,tf],y0);    %求数值解
y1=sqrt(t+1)+1;                   %求精确解
```

分别输出求解的数值解 y 和精确解 y1，对比如表 35-2 所示。

表 35-2　数值解 $y$ 和精确解 $y1$ 的对比

| y | y1 |
|---|---|
| 2.0000 | 2.0000 |
| 2.1490 | 2.1489 |
| 2.3495 | 2.3491 |
| 2.5239 | 2.5232 |
| 2.6803 | 2.6793 |
| 2.8234 | 2.8221 |
| 2.9561 | 2.9545 |
| 3.0805 | 3.0785 |
| 3.1978 | 3.1954 |
| 3.3093 | 3.3065 |
| 3.4157 | 3.4125 |
| 3.4529 | 3.4495 |

由表 35-2 中可以看出，数值解 y 和精确值 y1 两者近似。

**例 35-31**　求解常微分方程 $\frac{d^2y}{dt^2}-\mu(1-y^2)\frac{dy}{dt}+y=0$ 的解，其初值条件为 $\begin{cases} y(0)=0.8 \\ y'(0)=0 \end{cases}$，并画出解的图形。

解：这是一个二阶非线性方程，用现成的方法均不能求解，但可以通过下面的变换，将二阶方程化为一阶方程组，即可求解。

令：$x_1=y,x_2=\frac{dy}{dt},\mu=7$，则原方程化为：

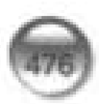
$$\begin{cases} \frac{dx_1}{dt}=x_2 \\ \frac{dx_2}{dt}=7(1-x_1^2)x_2-x_2 \end{cases}，且 \begin{cases} x_1(0)=0.8 \\ x_2(0)=0 \end{cases}$$

在 MATLAB 中求解，首先建立函数文件 vdp.m。

```
function fy=vdp(t,x)
fy=[x(2);7*(1-x(1)^2)*x(2)-x(1)];
```

再编写 m 文件求解微分方程。

```
y0=[0.8;0]
[t,x]=ode45(@vdp,[0,40],y0);
y=x(:,1);dy=x(:,2);
```



```
plot(t,y,t,dy)
```

输出的解的图形如图 35-3 所示。

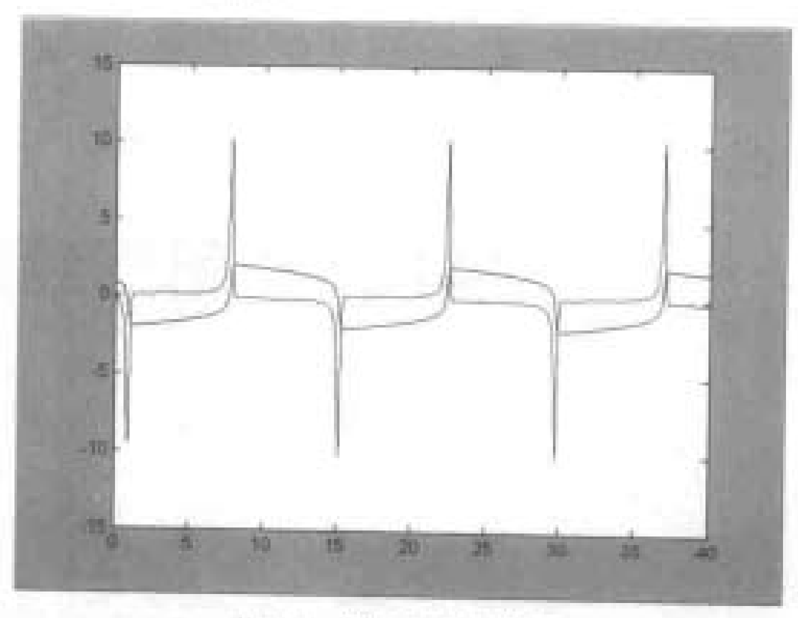



图 35-3 例 35-30 输出的解

## 35.8 小结

本章通过大量的实例，讲述了 MATLAB 在高等数学计算中的应用，包括极值运算、求导数、求积分、级数运算、函数的零点（即极值点）、代数方程求解和常微分方程求解，掌握这些方法是熟练应用 MATLAB 进行数学计算的基础。

# 第 36 章

# MATLAB 图形绘制实例

MATLAB 中具有丰富的二维图形和三维图形的绘制函数，这是其数据可视化技术的重要组成方面。MATLAB 正是通过丰富的图形函数，使得用户能够方便直观地查看和分析个人数据。

## 36.1 二维绘图

MATLAB 中可以绘制各种常用的一元函数图形，也可以对具有一元函数关系的用户采样数据进行绘图。

### 36.1.1 函数绘图

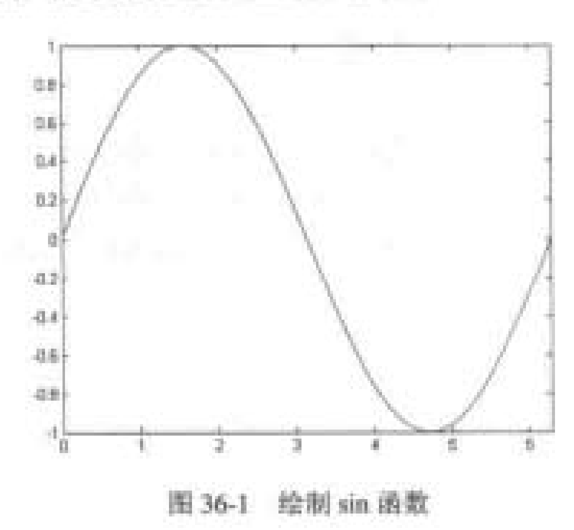


图 36-1 绘制 sin 函数

例 36-1 绘制 sin 函数。

解：在命令窗口中输入：

```
>> fplot(@sin,[0,2*pi])
```

结果如图 36-1 所示。

例 36-2 绘制匿名函数。

解：在命令窗口中输入：

```
>> fhd=@(x)(x.^2+1./x.^2);
>> fplot(fhd,[0.5 1.5])
```

结果如图 36-2 所示。

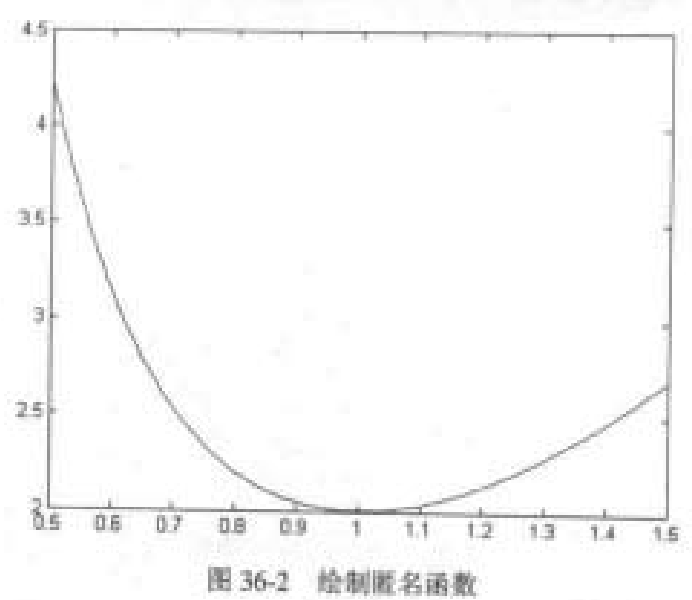



图 36-2　绘制匿名函数

**例 36-3**　简易绘制隐函数。

**解**：在命令窗口中输入：

```
>> ezplot('x.^3.*y+x.*y^3=6',[-5 5])
```

结果如图 36-3 所示。

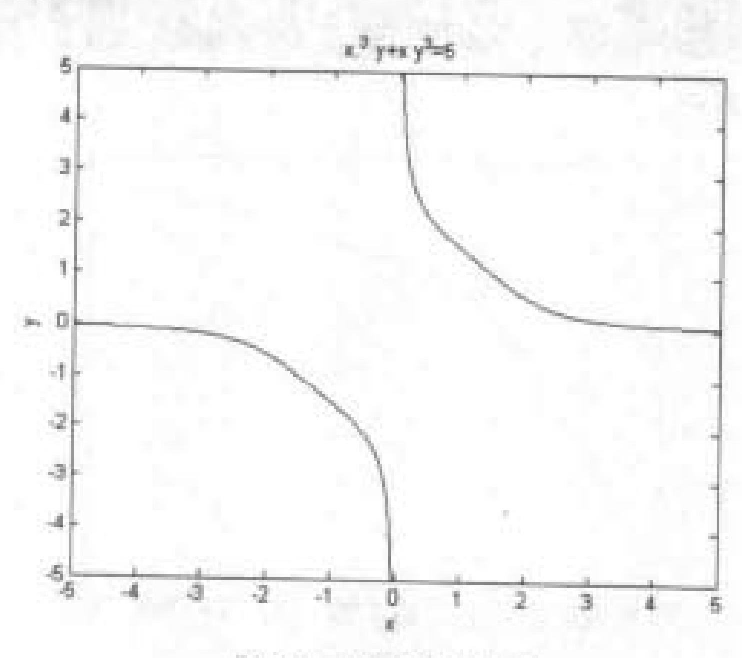



图 36-3　简易绘制隐函数

**例 36-4**　极坐标函数绘图。

**解**：在命令窗口中输入：

```
>> ezpolar('3*sin(t)-3*cos(t)',[-2 2])
```

结果如图 36-4 所示。

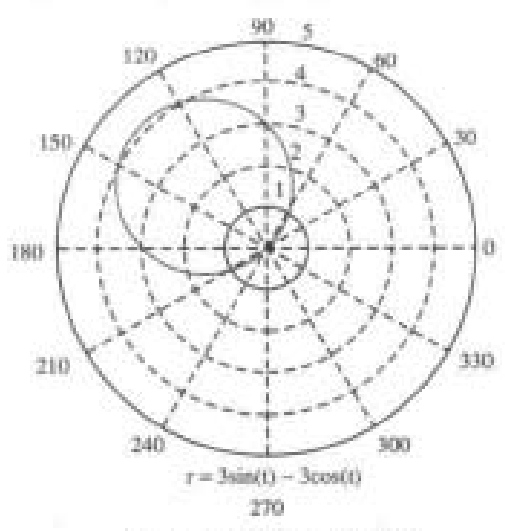



图 36-4　极坐标函数绘图

### 36.1.2　离散数据绘图

例 36-5　离散数据点直接绘图。

**解：**在命令窗口中输入：

```
>> x=sort(rand(1,10));
>> y=sort(rand(1,10));
>> plot(x,y)
```

结果如图 36-5 所示。

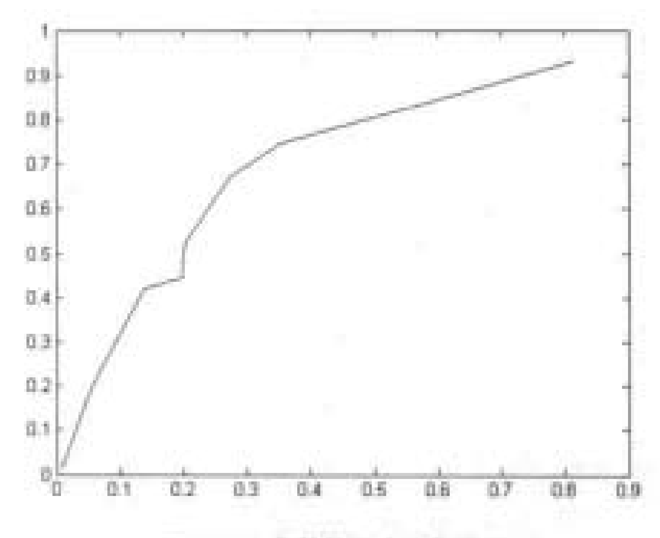



图 36-5　离散数据点直接绘图

例 36-6　离散数据点拟合绘图。

**解：**在命令窗口中输入：

```
>> x=sort(rand(1,10));
>> y=sort(rand(1,10));
```



```
>> p=polyfit(x,y,2)
p =
   -0.4969    1.3279   -0.0027
>> Y=polyval(p,x);
>> plot(x,y,'b*',x,Y,'g')
```

结果如图 36-6 所示。

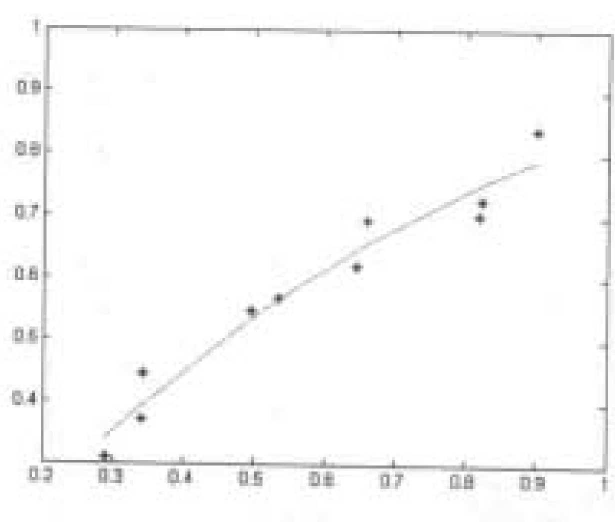


图 36-6　离散数据点拟合绘图

例 36-7　离散数据点插值绘图。

解：在命令窗口中输入：

```
>> x=sort(rand(1,10));
>> y=sort(rand(1,10));
>> X=linspace(0,1,100);
>> Y=spline(x,y,X);
>> plot(x,y,'b*',X,Y,'g')
```

结果如图 36-7 所示。

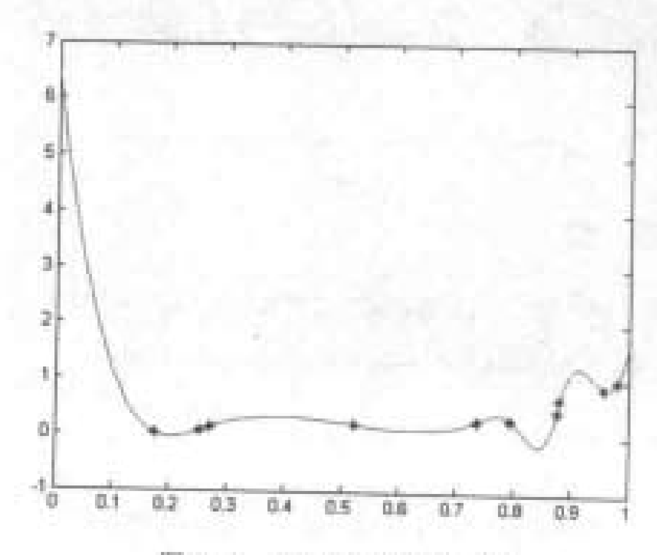


图 36-7　离散数据点插值绘图

### 36.1.3 特殊坐标轴绘图

例 36-8 双纵轴绘图。

**解**：在命令窗口中输入：

```
>> x=-1:0.1:1;
>> y=x.^3;
>> z=x.^2;
>> plotyy(x,y,x,z)
```

结果如图 36-8 所示。

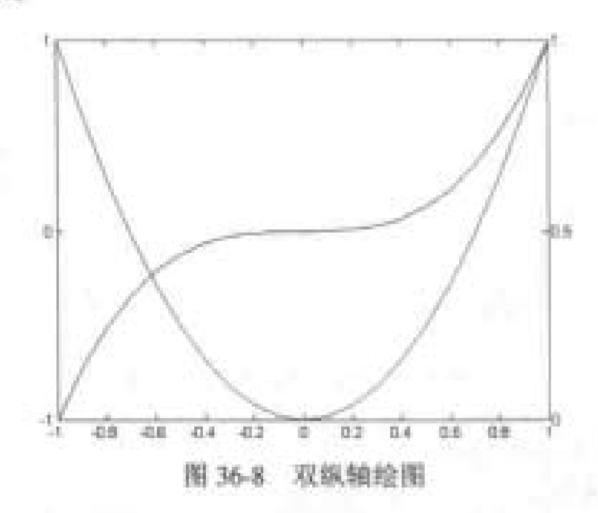


图 36-8 双纵轴绘图

例 36-9 半对数坐标轴绘图。

**解**：在命令窗口中输入：

```
>> x=-1:0.1:1;
>> y=exp(x).*cos(x);
>> subplot(2,1,1)
>> semilogy(x,y,'b-.')
>> subplot(2,1,2)
>> plot(x,y,'r-.')
```

结果如图 36-9 所示。

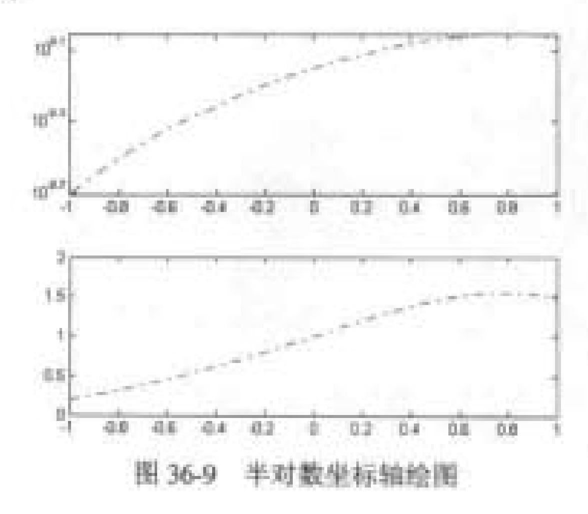


图 36-9 半对数坐标轴绘图

例36-10 双对数坐标轴绘图。

解：在命令窗口中输入：

```
>> x=0:100;
>> y=exp(x);
>> loglog(x,y)
```

结果如图36-10所示。

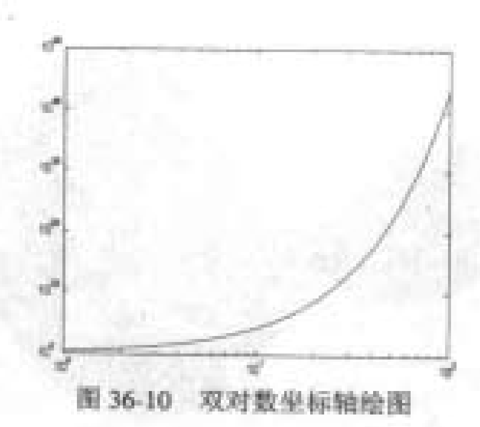


图36-10 双对数坐标轴绘图

## 36.2 三维绘图

### 36.2.1 二元函数绘图

例36-11 二元函数peaks绘图。

解：在命令窗口中输入：

```
>> peaks

z =  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2)...
   - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2)...
   - 1/3*exp(-(x+1).^2 - y.^2)

>> surf(peaks)
```

结果如图36-11所示。

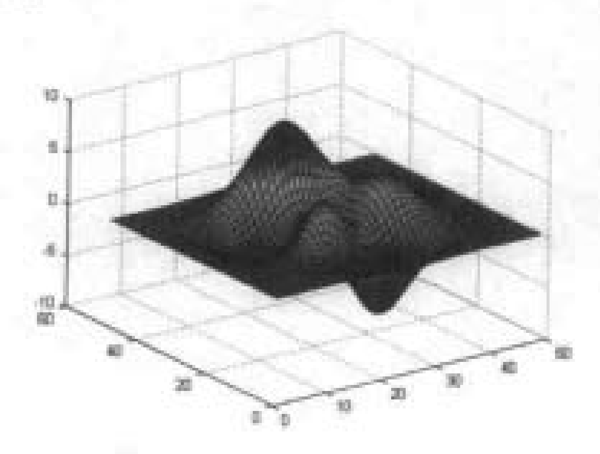



图36-11 二元函数peaks绘图

例 36-12　二元匿名函数绘图。

解：在命令窗口中输入：

```
>> ezsurf(@(x,y)(x.^2-y.^2),[-1 1 -1 1])
```

结果如图 36-12 所示。

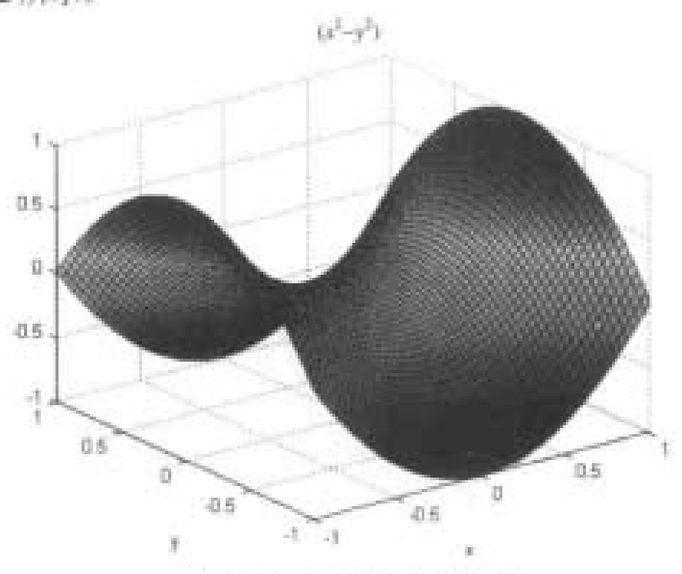


图 36-12　二元匿名函数绘图

### 36.2.2　三维曲线绘图

例 36-13　三维曲线绘图。

解：在命令窗口中输入：

```
>> t=0:2*pi;
>> x=cos(t).*sin(t);
>> y=2+t;
>> z=t.^2;
>> plot3(x,y,z)
```

结果如图 36-13 所示。

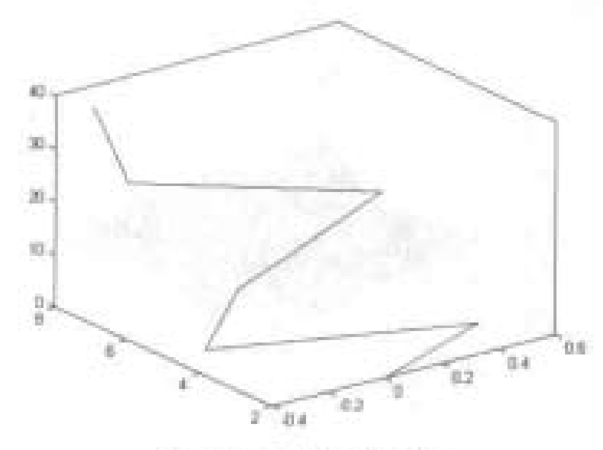


图 36-13　三维曲线绘图



### 36.2.3 三维曲面绘图

例36-14 三维曲面绘图。

解：在命令窗口中输入：

```
>> [t,s]=meshgrid(0:0.1*pi:2*pi);
>> x=sin(t).*cos(s);
>> y=cos(t).*sin(s);
>> z=cos(2*t).*cos(2*s);
>> subplot(1,2,1)
>> mesh(x,y,z)
>> subplot(1,2,2)
>> surf(x,y,z)
```

结果如图36-14所示。

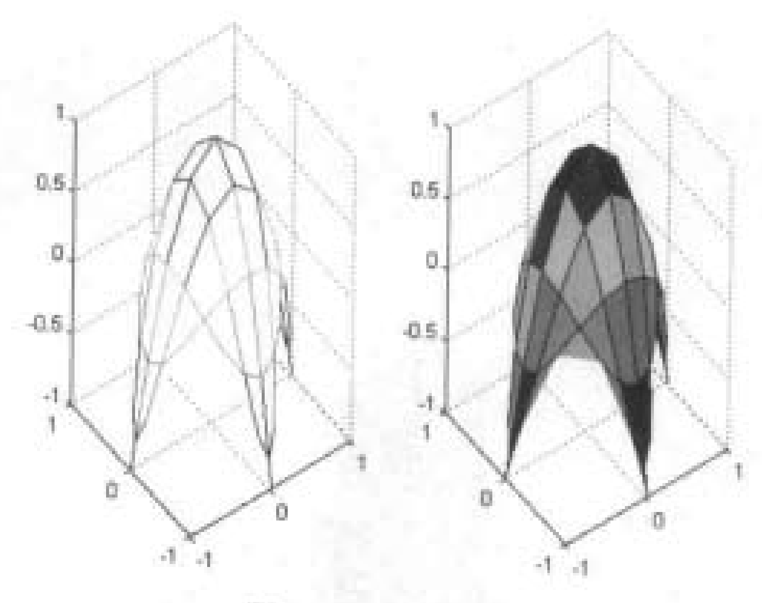



图36-14 三维曲面绘图

## 36.3 特殊分析用图

### 36.3.1 柱状图

例36-15 二维柱状图。

解：在命令窗口中输入：

```
>> x=rand(1,5)
x =
    0.5828    0.4235    0.5155    0.3340    0.4329
>> bar(x)
```

结果如图36-15所示。

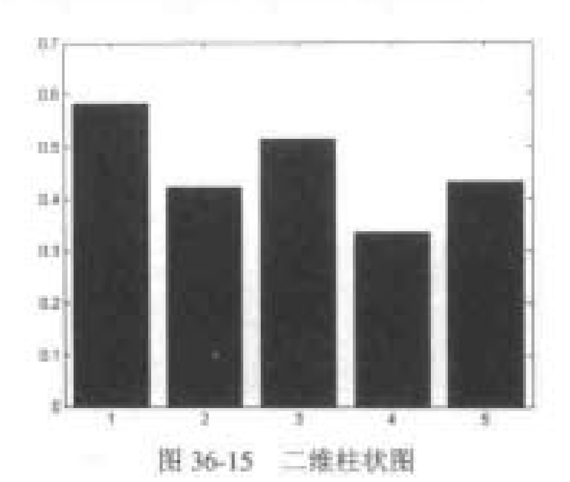


图 36-15　二维柱状图

例 36-16　三维柱状图。

**解**：在命令窗口中输入：

```
>> x=rand(3,5)
x =
    0.2259    0.5298    0.3798    0.4611    0.0592
    0.5798    0.6405    0.7833    0.5678    0.6029
    0.7604    0.2091    0.6808    0.7942    0.0503
>> bar3h(x,'stack')
```

结果如图 36-16 所示。

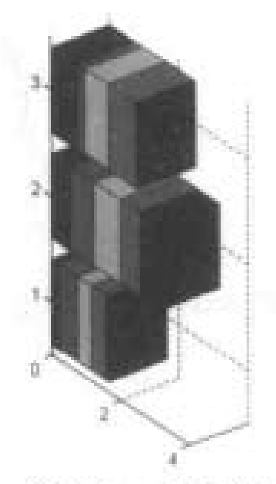


图 36-16　三维柱状图

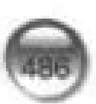
## 36.3.2　直方图

例 36-17　直方图。

**解**：在命令窗口中输入：

```
>> x=randn(1,1000);
>> hist(x,20)
```

结果如图 36-17 所示。



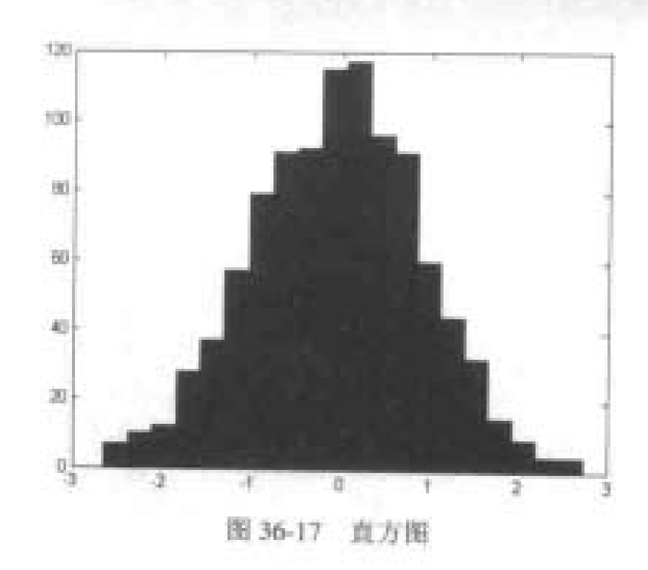


图36-17　直方图

### 36.3.3　饼图

**例36-18**　二维和三维饼图。

**解**：在命令窗口中输入：

```
>> x=rand(1,5)
x =
   0.2071    0.6072    0.6299    0.3705    0.5751
>> subplot(1,2,1)
>> pie(x)
>> subplot(1,2,2)
>> pie3(x,[0 1 0 1 0])
```

结果如图36-18所示。

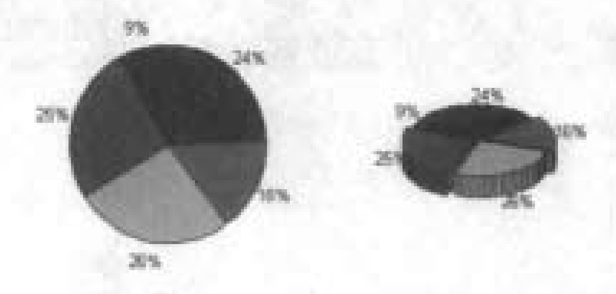


图36-18　二维和三维饼图

### 36.3.4　散点图

**例36-19**　二维和三维散点图。

**解**：在命令窗口中输入：

```
>> x=rand(1,15);
>> y=rand(1,15);
>> z=rand(1,15);
>> subplot(1,2,1)
>> scatter(x,y,'r*')
```

```
>> subplot(1,2,2)
>> scatter3(x,y,z,'bo')
```

结果如图 36-19 所示。

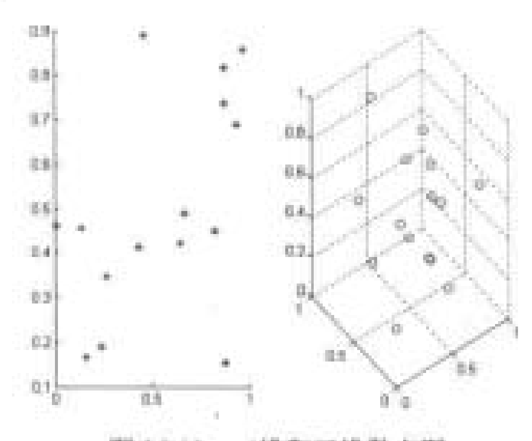


图 36-19　二维和三维散点图

### 36.3.5　等高线图

**例 36-20**　二维和三维等高线图。

**解**：在命令窗口中输入：

```
>> z=peaks;
>> subplot(2,2,1)
>> contour(z)
>> subplot(2,2,3)
>> contourf(z)
>> subplot(2,2,2)
>> contour3(z,20)
>> subplot(2,2,4)
>> surfc(z)
```

结果如图 36-20 所示。

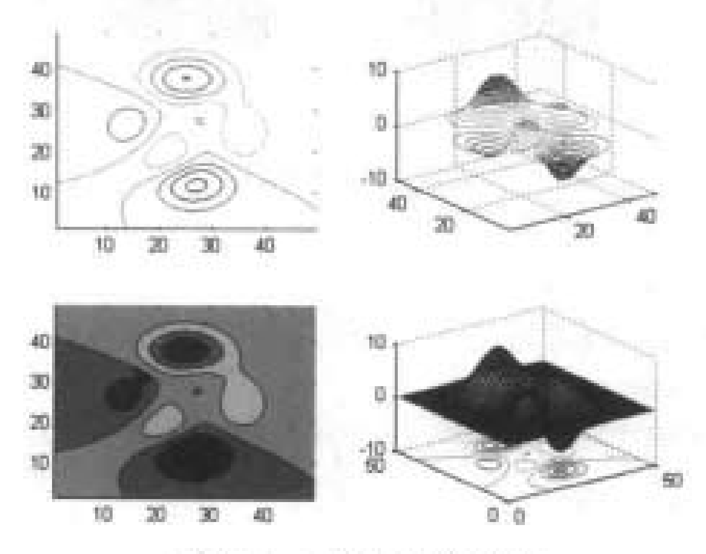



图 36-20　二维和三维等高线图



## 36.4 小结

本章通过大量的实例，讲述了不同情况下 MATLAB 图形绘制的函数，包括二维图形、三维图形和特殊分析图形。掌握这些不同情况下 MATLAB 绘图函数和方法，是应用 MATLAB 实现数据可视化的基础。



# 第 37 章 MATLAB 扩展编程实例

MATLAB 具有强大的计算绘图功能，以及大量完善的算法，与其他高级开发语言实现扩展编程，例如与 VC++结合，充分利用两者的优势是 MATLAB 的研究与应用的一个热点。

## 37.1 MATLAB 与 VC++混合编程概述

### 37.1.1 混合编程的背景

从前面章节可以看出，MATLAB 是针对数值计算的交互式软件，其强大的数据处理能力和丰富的工具箱使得编程变得简单，应用程序开发周期大幅度缩短，编程效率高。

同时 MATLAB 是一种解释性的语言开发平台，它的执行效率比较低。在很多时候仅采用MATLAB 还是不能很好地实现某些功能，许多其他语言编写的算法程序若要在MATLAB 环境中运行，还需要重新编写程序。

Visual C++是 Windows 平台下强有力的高级编程语言，能够方便快速地开发出界面友好，执行速度快，易于维护升级的软件系统。然而 Visual C++只提供了一些基本的数学函数库，当遇到复杂的数值运算时，重新编写程序代码延长软件开发周期，增加软件开发成本。

MATLAB 拥有独立的数学函数库，包含有大量优化了的数学函数，同时提供了对 C 和 C++等语言的函数接口，用户可以方便地在 VC++的集成开发环境中调用。但 MATLAB 的应用程序接口并不是很强大，它不能传输除了数字之外的其他数据，而 VC++却具有强大的程序接口，能传输任何数据，但其进行复杂计算的能力不是很强。因此，若将两者结合起来，协同工作，必将提高软件开发效率。

MATLAB 7.x 提供了多种与 VC++的混合编程的方式，为科研与工程开发提供更为强大的技术支持。

### 37.1.2 混合编程的方式

VC++与 MATLAB 的混合编程方式一般有如下几种。

（1）MEX 文件。

（2）MAT 文件应用程序。

（3）使用 Matcom 方式。使用 Matcom 编译器可以将 MATLAB 源代码译成同等功能的 C++代码，既保持了 MATLAB 的优良算法，又提高了执行速度，它还支持一定的图形显示，生成代码的可读性很好，简单便捷，功能强大，应用灵活。

（4）使用 MATLAB 引擎。通过 MATLAB 引擎，采用客户机/服务器的计算模式。在 VC++中设计程序框架，作为前端客户机，通过调用 MATLAB 引擎与后台 MATLAB 服务器建立连接，实现命令和数据信息的传递。

（5）使用 MATLAB 的 C/C++编译器 mcc。采用 MATLAB 的 C/C++编译器 mcc 将.m 源文件转化为 C/C++等各种不同类型的源代码，并在此基础上根据应用需要生成 MEX 文件、独立可执行应用程序等文件类型，大大提高程序的运行速度，以及代码的执行效率。这种方法主要使用 mcc 命令实现文件的转化。

（6）使用 COM。利用 COM Builder 工具将 MATLAB 程序编译成二进制 COM 组件，它实际上是以前编译 C/C++文件的完美扩展，简单高效地解决了以前混合编程中存在的问题。

下面将通过实例，重点对应用较广的（4）、（5）和（6）这三种混合编程的方式进行介绍。

## 37.2 使用 MATLAB 引擎

MATLAB 将数值分析、矩阵计算、信号处理和图形显示结合在一起，包含大量高度集成的函数可供调用，是适合科学研究、工程设计等众多学科领域使用的一种简洁、高效的编程工具。基于 VC++和 MATLAB 混合编程最简单也最直接的方法就是调用 MATLAB 引擎。以下部分将详细介绍通过 VC++ 6.0 调用 MATLAB 7.0 引擎来达到 VC++与 MATLAB 数据共享编程的方法。

### 37.2.1 MATLAB 引擎

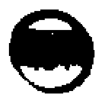
#### 1. MATLAB 引擎基础

所谓 MATLAB 引擎(engine)，是指一组 MATLAB 提供的接口函数，支持 C/C++，Fortran 等语言，通过这些接口函数，用户可以在其他编程环境中实现对 MATLAB 的控制。其主要功能有：

（1）打开/关闭一个 MATLAB 对话；

（2）向 MATLAB 环境发送命令字符串；

（3）从 MATLAB 环境中读取数据；





（4）向 MATLAB 环境中写入数据。

通过引擎方式，应用程序可以：

（1）打开一个新的 MATLAB 进程，可以控制它完成任何计算和绘图操作；

（2）对所有的数据结构提供 100%的支持；

（3）引擎方式打开的 MATLAB 进程，会在任务栏显示自己的图标，打开该窗口，可以观察主程序通过 engine 方式控制 MATLAB 运行的流程，并可在其中输入任何 MATLAB 命令。

与其他各种接口相比，引擎所提供的 MATLAB 功能支持是最全面的。

实际上，通过引擎方式建立的对话，是将 MATLAB 以 ActiveX 控件方式启动的。在 MATLAB 初次安装时，会自动执行一次 matlab/regserver，将自己在系统的控件库中注册。

**2. 数据类型 mxArray**

MATLAB 引擎函数中，所有与变量有关的数据类型都是 mxArray 类型。数据结构 mxArray 以及大量的 mx 开头的函数，广泛用于 MATLAB 引擎程序和 MATLAB C 数学库中。mxArray 是一种很复杂的数据结构，与 MATLAB 中的 array 相对应。下面讲述 MATLAB 的 mxArray 类型，并介绍几个常用的 mxArray 函数。

需要注意的是，在使用 mxArray 类型的程序中，应包含头文件 matrix.h，由于在引擎程序中，一般会包含头文件 engine.h，该文件里面已经包含了 matrix.h，因此无需重复包含。

（1）mxArray 数据的创建和删除

MATLAB 有很多种变量类型，但它们都有相同的 mxArray 数据结构，对应于每种类型，基本上都有一个创建函数。

① 数组的建立采用 mxCreatexxx 形式的函数，其中 xxx 表示数据类型，例如创建一个 double 类型数组的函数是 mxCreateDoubleMatrix，函数形式如下：

mxArray *mxCreateDoubleMatrix(int *m*, int *n*, mxComplexity ComplexFlag);

其中，参数 *m* 和 *n* 为矩阵的行数和列数；ComplexFlag 为常数，用来区分矩阵中元素是实数还是复数，取值分别为 mxREAL 和 mxCOMPLEX。

类似的创建函数还有 mxArray *mxCreateString(const char *str)，用来创建一个 string 类型的串。

② 数组的删除函数为 mxDestroyArray，该函数声明如下：

void mxDestroyArray(mxArray *array_ptr);

其中，参数 array_ptr 为要删除的数组指针。一般地，在 VC++与 MATLAB 交互中，以上两种类型是常用的类型，由于篇幅关系，其他类型数组的创建在此不作介绍。

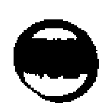
（2）mxArray 数据属性的操作

① mxArray 数据的大小

使用 mxGetM 和 mxGetN 函数可以获得 mxArray 数组每一维上元素的个数，其中 mxGetM 用来获得数组第一维的元素个数，对于矩阵来说就是行数。mxGetM 函数声明如下：

int mxGetM(const mxArray *array_ptr);

其中，返回的 array_ptr 对应数组第一维的元素个数（行数）。





mxGetN 函数声明如下：

int mxGetN(const mxArray *array_ptr);

其中，返回 array_ptr 对应数组其他维的元素个数，对于矩阵来说是列数。对于多维数组来说是从第二维到最后一维的各维元素个数的乘积。

mxGetDimensions 函数可以获得某一特定维的元素个数，该函数声明如下：

const int *mxGetDimensions(const mxArray *array_ptr);

其中，函数返回 array_ptr 各维的元素个数保存在一个 int 数组中返回。对于常用的矩阵来说，用 mxGetM 和 mxGetN 两个函数就可以了。

mxGetNumberOfDimensions 函数可以获得数组总的维数，该函数声明如下：

int mxGetNumberOfDimensions(const mxArray *array_ptr);

该函数结合 mxSetM、mxSetN 可以设置矩阵的行数和列数。mxSetM 函数说明如下：

void mxSetM(mxArray *array_ptr, int *m*);

该函数设置数组为 *m* 行。

mxSetN 函数说明如下：

void mxSetN(mxArray *array_ptr, int *n*);

该函数设置数组为 *n* 列。

② 获得 mxArray 数组数据的实部和虚部数据指针

对于常用的 double 类型的数组，可以用 mxGetPr 和 mxGetPi 两个函数分别获得其实部和虚部的数据指针，这两个函数的声明如下：

double *mxGetPr(const mxArray *array_ptr);

该函数返回数组 array_ptr 的实部指针。

double *mxGetPi(const mxArray *array_ptr);

该函数返回数组 array_ptr 的虚部指针。

利用这两个函数，就可以对 mxArray 类型的数组中的数据进行读写操作。

### 3. 引擎函数

在 MATLAB 7 的引擎函数库中，总共提供了 10 个 C 语言的引擎函数，它们分别为：

（1）engClose

（2）engEvalString

（3）engGetVariable

（4）engSetVariable

（5）engGetVisible

（6）engOpen

（7）engOpenSingleUse

（8）engOutputBuffer

（9）engPutVariable

（10）engSetVisible





MATLAB 7 中，把以前老版本中的 9 个函数淘汰了，这些函数分别为：

（1）engGetArray

（2）engGetFull

（3）engGetMatrix

（4）engPutArray

（5）engPutFull

（6）engPutMatrix

（7）engSetEvalCallback

（8）engSetEvalTimeout

（9）nengWinInit

下面对 VC++中调用的引擎函数分别进行介绍。

（1）engOpen

engOpen 函数用来打开 MATLAB engine，该函数声明如下：

```
Engine *engOpen(const char *startcmd);
```

其中，参数 startcmd 是用来启动 MATLAB 引擎的字符串参数。在 Windows 操作系统中只能为 NULL；函数返回值是一个 Engine 类型的指针，它是在 engine.h 中定义的 engine 数据结构。

（2）engClose

engClose 函数用来关闭 MATLAB 引擎，该函数声明如下：

```
int engClose(Engine *ep);
```

其中，参数 ep 代表要被关闭的引擎指针。函数返回值为 0 表示关闭成功，返回 1 表示发生错误。

（3）engEvalString

engEvalString 函数用来向 MATLAB 发送命令字符串，也就是发送命令让 MATLAB 执行，该函数声明如下：

```
int engEvalString(Engine *ep, Const char *string);
```

其中，参数 ep 为函数 engOpen 返回的引擎指针；字符串 string 为要 MATLAB 执行的命令；函数返回值为 0 表示成功执行，返回 1 说明执行失败（表示命令不能被 MATLAB 正确解释或 MATLAB 引擎已经关闭了）。

（4）engOutputBuffer

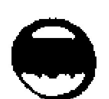
engOutputBuffer 函数用来获取 MATLAB 命令窗口的输出，该函数声明如下：

```
int engOutputBuffer(Engine *ep, char *p, int n);
```

其中，参数 ep 为 MATLAB 引擎指针；p 为用来保存输出结构的缓冲区；n 为保存的最大字符个数，通常就是缓冲区 p 的大小。

该函数执行后，接下来的 engEvalString 函数所引起的命令行输出结果，会在缓冲区 p 中保存。如果要停止保存，只需调用代码 engOutputBuffer(ep, NULL, 0)。

（5）engPutVariable



engPutVariable 函数用来写 MATLAB 数据，向 MATLAB 引擎工作空间写入变量，该函数声明如下：

```
int engPutVariable(Engine *ep, const char *name, const mxArray *mp);
```

其中，参数 ep 为打开的 MATLAB 引擎指针；mp 为指向被写入变量的指针；name 为变量写入后，在 MATLAB 引擎工作空间中的变量名。函数返回值为 0 表示写入变量成功，返回值为 1 表示发生错误。

（6）engGetVariable

engGetVariable 函数用来读 MATLAB 数据，从 MATLAB 引擎工作空间中获取变量，该函数声明如下：

```
mxArray *engGetVariable(Engine *ep, const char *name);
```

其中，参数 ep 为打开的 MATLAB 引擎指针；name 为以字符串形式指定的数组名；函数返回值是指向 name 数组的指针，类型为 mxArray*。

（7）engSetVisible

engSetVisible 函数用来设置调用引擎时显示或隐藏MATLAB 主窗口，该函数声明如下：

```
int engSetVisible(Engine *ep, bool value);
```

其中，参数 ep 为打开的 MATLAB 引擎指针，value 为是否显示的标志，取值 true（或 1）表示显示 MATLAB 窗口，取值 false（或 0）表示隐藏 MATLAB 窗口。函数返回值为 0 表示设置成功，为 1 表示有错误发生。

默认情况下，以 engine 方式调用 MATLAB 的时候，会打开 MATLAB 主窗口，可在其中随意操作，但有时也会干扰应用程序的运行，可用以下设置是否显示该窗口。

（8）engGetVisible

engGetVisible 函数用来获取调用引擎时 MATLAB 主窗口显示或隐藏的属性，该函数声明如下：

```
int engGetVisible(Engine *ep, bool *value);
```

其中，参数 ep 为打开的 MATLAB 引擎指针，value 为用来保存显示/隐藏情况的变量（采用指针方式传递），函数返回值为 0 表示获取成功，为 1 表示有错误发生。

### 37.2.2 编程实例

下面通过一个简单的利用 VC++调用 MATLAB 画图的程序实例，讲述 MATLAB 引擎的使用方法。读者可以充分利用 MATLAB 强大的数据读写、显示能力和 VC++编程的高效率。

#### 1. VC++中的设置

在 VC++中要能调用 MATLAB 引擎，首先必须包含引擎头文件 engine.h 并引入 MATLAB 对应的库文件 libmx.lib，libmat.lib，libeng.lib，然后在 VC++中打开工程后，进行如下设置。

（1）通过菜单工程/选项，打开选项(Options)页，进入 Directories 页面，在目录下拉列表框中选择 Include files，添加路径："D:\matlab7\extern\include"（假定 MATLAB 安装在



d:\matlab7 目录），如图 37-1 所示。

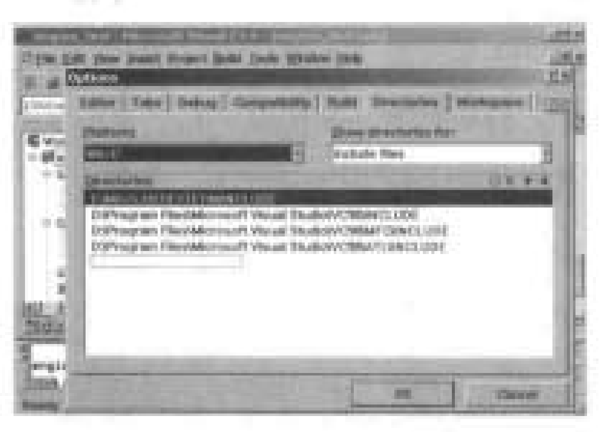


图 37-1　Options 设置页面

（2）选择 Library files，添加路径：D:\MATLAB7\extern\lib\win32\microsoft\msvc60 和 D:\MATLAB7\extern\lib\win32。

（3）通过菜单工程/设置，打开工程设置属性页，进入 Link 页面，在 Object/library modules 编辑框中，添加文件名 libmx.lib，libmat.lib，libeng.lib。

以上步骤（1）、（2）只需设置一次，而步骤（3）对每个工程都要单独设定，对于其他 C++编译器如 Borland C++builder，设置大体相同，此处不再赘述。

此外，在调用 MATLAB 引擎之前，在相关文件中需要包含引擎的头文件 engine.h，格式为：

#include "engine.h"

该文件包含了引擎 API 函数的说明和所需数据结构的定义。

### 2．程序实现

实例演示如何利用 VC++调用 MATLAB 绘图，程序的主要功能是在 VC++中对数组 $x$ 计算函数值 $y=\cos(x)$，然后调用 MATLAB 绘制 $y$ 对 $x$ 的图形。

在 VC++中新建工程，编写代码如下：

```
#include "stdafx.h"
#include <iostream>
#include <math.h>
//包含引擎头文件 engine.h
#include "engine.h"
using namespace std;
void main()
{
    const int N=50;
    double  x[N],y[N];
    int j = 1;
    int i;
    for(i=0;i<N;i++)
{
//计算数组 x 和 y
        x[i]=i+1;
y[i]=cos(x[i]);
```



```
    j*=-1;
    }
    //定义 MATLAB 引擎指针。
    Engine *ep;
    //测试是否启动 MATLAB 引擎成功。
    if(!(ep=engOpen(NULL)))
    {
            cout<<"Can't start MATLAB engine!"<<endl;
            exit(1);
        }
    //定义 mxArray，为 1 行，N 列的实数数组。
    mxArray *xx=mxCreateDoubleMatrix(1,N, mxREAL);
    mxArray *yy = mxCreateDoubleMatrix(1,N, mxREAL);
    //将数组 x 复制到 mxarray 数组 xx 中。
    memcpy(mxGetPr(xx), x, N*sizeof(double));
    //将数组 x 复制到 mxarray 数组 yy 中。
    memcpy(mxGetPr(yy), y, N*sizeof(double));
    //将 mxArray 数组 xx 写入到 MATLAB 工作空间，命名为 xx。
    engPutVariable(ep, "xx",xx);
    //将 mxArray 数组 yy 写入到 MATLAB 工作空间，命名为 yy。
    engPutVariable(ep, "yy",yy);
    //向 MATLAB 引擎发送画图命令。
    engEvalString(ep, "plot(xx, yy); ");
    //销毁 mxArray 数组 xx 和 yy。
    mxDestroyArray(xx);
    mxDestroyArray(yy);
    cout <<"Press any key to exit!" <<endl;
    cin.get();
    //关闭 MATLAB 引擎
    engClose(ep);
    }
```

编译并运行程序，程序运行需要一段时间，这是由于在 VC++中启动通过引擎 MATLAB 需要时间，运行后输出如图 37-2 所示的结果。

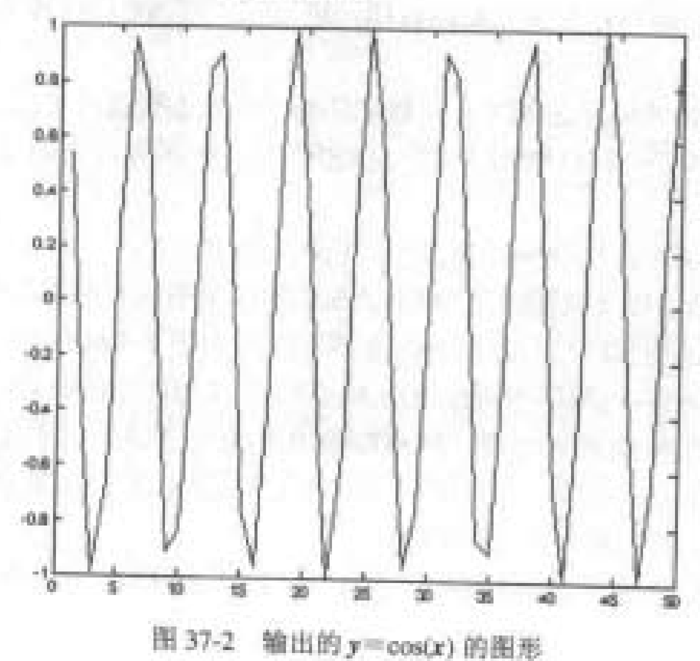
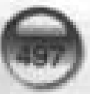



图 37-2　输出的 $y=\cos(x)$ 的图形



可以查看在 MATLAB 工作空间中的变量，在启动的 MATLAB 命令窗口中输入：

```
who
%输出为
Your variables are:
xx  yy
```

可见工作空间中存在这两个变量 xx 和 yy。

## 37.3 使用 mcc 编译器

### 37.3.1 mcc 编译器

MATLAB 在许多学科领域中成为计算机辅助设计与分析、算法研究和应用开发的基本工具和首选平台。但由于 MATLAB 的编译器采用伪编译的方式，在 MATLAB 中编写的程序无法脱离其工作环境而独立运行。针对这个问题，Mathworks 公司为 MATLAB 提供了应用程序接口，允许 MATLAB 和其他应用程序进行数据交换，并且提供了 C/C++数学和图形函数库，为在其他程序设计语言调用 MATLAB 高效算法提供了可能。

C++语言是新一代的以面向对象（OOP）概念为根本的高级程序设计语言，它的面向对象的概念更加符合程序员开发软件的思维习惯，类封装性和模块化的构造非常适合软件的移植和维护。使用 C++开发有助于提高软件工程的质量，VC++与其他一些 C++编译器相同，都是以 C++语言为编译对象。

7.x 版本的 MATLAB 软件包中提供了 C/C++的数学和图形库，通过其编译器的支持，MATLAB 中编写的.m 文件可以转换成以 C/C++代码编写的文件，而且用户可以将 m 文件生成 DLL 库，甚至可以直接调用其中的库函数，生成并发布不必依赖 MATLAB 环境的可执行文件。通过 mcc 编译器生成 C/C++代码，进而可以在 VC++或者其他编译器生成可独立执行的应用程序。

使用 MATLAB 的 Compiler 将*.m 函数文件编译为动态链接库 DLL，然后通过在 VC++中调用该 DLL 实现 VC++和 MATLAB 的混合编程，这是一种方便快捷的 VC++和 MATLAB 混合编程方法。

但同时，MATLAB 的 Compiler 也具有一定的局限性：

（1）混合编程的过程中只能使用 MATLAB 数学库中的函数和图形库中的部分函数；

（2）MATLAB 的图形库只有在用 mcc 生成可执行程序（*.exe）时才能引用；

（3）不支持在 VC++的程序中直接使用 MATLAB 图形库中的函数；

（4）使用 mcc 生成 DLL 的时候，MATLAB 图形库中的函数也不是能全部通过测试；

（5）某些工具箱中的函数，如果用到一些数学库之外的函数，就有可能出现错误，即使编译通过，在运行的时候也有可能出错。

随着 MATLAB 的不断升级，Mathworks 推出了 MATLAB COM Builder 创建 COM 组件，对这种不兼容性有了一定的改进，使得 MATLAB 和高级语言的混合编程变得越来越实用且简单。



## 37.3.2 MATLAB的设置及创建动态链接库

下面介绍如何利用MATLAB提供的C/C++编译器，将m文件编译成可执行的应用程序。下面用的编译环境是：MS VC++6.0和MATLAB7.0。

### 1. MATLAB编译环境设置

使用MATLAB的编译器将*.m函数文件编译为动态链接库DLL之前，必须对MATLAB的环境进行相应的配置。具体步骤如下。

步骤1：在MATLAB的命令窗口中输入命令mbuild –setup，并根据MATLAB的提示选择合适的编译器，该过程如下所示。

```
>> mbuild -setup
%输出为
Please choose your compiler for building standalone MATLAB applications:
Would you like mbuild to locate installed compilers [y]/n? y
Select a compiler:
[1] Lcc C version 2.4 in D:\MATLAB7\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in D:\Program Files\Microsoft Visual
Studio
[0] None
%选择编译器的类型
Compiler: 2
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
Location: D:\Program Files\Microsoft Visual Studio
%确认
Are these correct?([y]/n): y
Try to update options file: C:\Documents and Settings\Administrator\
Application Data\MathWorks\MATLAB\R14\compopts.bat
From template:          D:\MATLAB7\BIN\WIN32\mbuildopts\msvc60compp.bat

Done . . .
--> "D:\MATLAB7\bin\win32\mwregsvr D:\MATLAB7\bin\win32\mwcomutil.dll"
DllRegisterServer in D:\MATLAB7\bin\win32\mwcomutil.dll succeeded
--> "D:\MATLAB7\bin\win32\mwregsvr D:\MATLAB7\bin\win32\mwcommgr.dll"
DllRegisterServer in D:\MATLAB7\bin\win32\mwcommgr.dll succeeded
```

然后输入命令mex -setup，

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n? y
Select a compiler:
[1] Lcc C version 2.4 in D:\MATLAB7\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in D:\Program Files\Microsoft Visual
Studio
[0] None
%选择编译器的类型
Compiler: 2
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
```

```
    Location: D:\Program Files\Microsoft Visual Studio
    Are these correct?([y]/n): y
    Try to update options file: C:\Documents and Settings\Administrator\
Application Data\MathWorks\MATLAB\R14\mexopts.bat
    From template:           D:\MATLAB7\BIN\WIN32\mexopts\msvc60opts.bat
    Done . . .
```

采用类似的方法配置编译器。

2．创建动态链接库（DLL）

在 MATLAB 中，创建动态链接库主要有两种方法。

（1）一种是使用 MATLAB 为 VC++ IDE 提供的 Add-in，这种方法比较简单，方便快捷。只要在 VC++中创建工程时选择 MATLAB Project Wizard，并且在接下来的步骤中的 Visual MATLAB Application Type 选择 Shared M-DLL 就可以了，然后添加*.m 文件，就可进行编译。

（2）另一种方法就是使用 MATLAB 的 mcc 命令将*.m 文件编译为动态链接库(*.DLL)。因为 Add-in 也是调用 Compiler 的命令 mcc 进行编译工作的，而且有时候这个 Add-in 还会出现不能使用的情况，因此本书主要讨论使用 mcc 命令的方法。

mcc 有很多参数可以使用，而且有多种用法，如表 37-1 所示。

表 37-1　mcc 编译器的参数

| 选　项 | 描　述 | 备　注 |
|---|---|---|
| a　filename | 把 filename 添加到 CTF 文件中 | |
| B | 产生 Excel 兼容的公式函数 | 要求有支持 Excel 的 MATLAB 编译器 |
| B　filename [:arg[,arg]] | 用 filename 的内容替换 mcc 命令行中的-B filename | 文件应该只包含 mcc 命令行选项，这些是 MathWorks 包括的选项文件：<br>-B csharedlib:foo　C shared library<br>-B cpplib:foo　C++ library |
| c | 产生 C wrapper 代码 | 相当于-T codegen |
| d directory | 输出到指定的目录（directory） | |
| f　filename | 调用 mbuild 时，使用指定的选项文件，文件名 | 建议使用 mbuild -setup |
| g | 产生调试信息 | |
| G | 仅调试，仅仅打开调试，因此包括了调试符号信息 | |
| I directory | 给 m 文件添加搜索目录 | 从 MATLAB 运行时，MATLAB 路径是自动包括的，但当从 DOS/UNIX 运行时，不是自动包括的 |
| l | 创建函数库的宏 | 相当于 -W lib -T link:lib |
| m | 用来产生一个 C 独立应用的宏 | 相当于 -W main -T link:exe |
| M string | 传递 string 给 mbuild | 用来定义编译时间选项 |
| N | 清除所有路径，但目录必需的最小路径集除外 | |
| o outputfile | 指定最终可执行性文件的名称/位置 | 添加适当的扩展 |
| P directory | 在顺序敏感的环境中，给编译路径添加目录（directory） | 要求-N 选项 |



续表

| 选　项 | 描　述 | 备　注 |
| --- | --- | --- |
| R option | 为MCR指定运行选项 | option = nojvm<br>nojit |
| T target | 指定输出阶段 | target = codegen<br>compile:bin<br>link:bin<br>where bin =exe<br>lib |
| v | Verbose：显示编译阶段 | |
| w option | 显示警告信息 | option = list<br>level<br>level:string<br>where level =disable<br>enable<br>error<br>No w option displays only serious warnings (default). |
| W type | 控制产生函数 wrappers | type = main<br>lib:<string><br>none<br>com:compname,<br>clname,version |
| Y licensefile | 当检查编译器的 license 时，使用 licensefile | |
| z path | 指定库和包括的文件的路径 | |
| ? | 显示帮助信息 | |

此处使用下列命令来创建动态链接库 DLL 文件：

mcc -B csglsharedlib:youlibname function1 funcion2…

其中 参数-B 表示使用的是 Bundle Files 作为参数；csglsharedlib 是-B 的参数，意思是生成使用 MATLAB 图形库的 C 共享动态链接库；youlibname 是要生成的动态链接库 DLL 的文件名；注意 csglsharedlib 与 youlibname 之间有一个冒号":"连接，function1 为要加入到动态链接库 DLL 中的函数名，如果有多个函数，各个函数名之间用空格隔开。

## 37.3.3 编程实例

### 1. MATLAB 程序实现

首先创建所需的 MATLAB 函数文件 mcc_test.m，该函数的功能是绘制一条正弦曲线，其代码如下所示。

```
function y=mcc_test(t)
t=0:0.001:t;
y=sin(2*pi*50*t);
```



```
plot(y);
```

然后使用命令 mcc -B csglsharedlib:mylib mcc_test 生成动态链接库 DLL。MATLAB 会生成一系列文件，有 mylib.c，mylib.exports，mylib.h，mylib.ctf，mylib_mcc_component_data.c，mylib.dll，mylib.exp，mylib.lib，其中 mylib.h mylib.lib mylib.dll 以及在 D:\MATLAB7\work\com_test\distrib\com_test_mcr\toolbox\compiler\deploy 目录下的 FigureMenuBar.fig 和 FigureToolBar.fig 是在以后所需要使用的。

2．VC++程序实现

为了能在 VC++中使用上面生成的*.DLL，首先要对 VC++的环境进行一些设置。

（1）设置 Include 和 Library 目录

在 VC++ IDE 中选择 Tools->Options->Directories。在 Show directories for：中选择 Include files，添加如下一个目录：

<MATLAB>\extern\include\

在 Show directoris for：中选择 Library files，添加如下两个目录：

<MATLAB>\extern\lib\win32

<MATLAB>\extern\lib\win32\microsof\msvc60

这里假设<MATLAB>为读者的 MATLAB 的安装目录。这些操作只需要一次，VC++ IDE 就会自动记录，自动应用到每一个工程（Project）。

（2）工程本身的一些设置

在 VC++ IDE 中选择 Project->Setting->C/C++，在 Category 中选择 Code Generation，在 Use run-time library 中选择 Multithreaded DLL。

在 Category 中选择 General，在 Excecutable for debug session：中添加 D:\Program Files\Microsoft Visual Studio\MyProjects\testmylib\Debug\testmylib.exe。

在 VC++ IDE 中选择 Project->Settings->Link，在 Output file name：中添加 Debug/testmylib.exe，在 Object/library modules：中填入 mylib.lib libmx.lib。

（3）为工程添加相应的文件

把刚才生成的 mylib.h，mylib.dll，mylib.lib 和<MATLAB>\extern\lib\win32 下的 libmatpm.lib，<MATLAB>\extern\lib\win32\microsof\msvc6 下的所有*.lib 文件复制到 VC++工程的文件夹下，并且用 add files to project…添加至工程中。

下面就可以使用刚才生成的动态链接库 mylib.dll 了。首先要在使用 DLL 中函数的源文件中加上头文件 mylib.h。在使用之前还需要对 mylib.dll 注册，使用完毕之后需要对其进行释放。注册时，使用函数 mylibInitiallize()；释放时，使用函数 mylibTerminate()。函数名的规则就是 youlibname+Initialize(Terminate)，这两函数在 mylib.h 中可以找到。mylibInitialize()不仅注册了刚才生成的 mylib.dll，同时它也注册了 MATLAB 本身的一些动态链接库（*.DLL）；同理 mylibTerminate()不仅释放了刚才生成的 mylib.dll，同时它也释放了 MATLAB 本身的一些动态链接库（*.DLL）。因此在注册了 mylib.dll 之后，也可以使用 MATLAB 的数学库中的函数，例如 mlfabs()，mlfAssign()等等。

在使用 mylib.dll 时，存在着 VC++和 mylib.dll 中的函数之间数据交互的问题。由于 mylib.dll 中的函数所接受的参数是 mxArray 型数据，因此要使用 MATLAB 提供的 External Interfaces/ API 函数来创建 mxArray 型数据，大量的 API 函数可以参见 MATLAB 的帮助文件 MATLAB\External Interfaces\API Reference\C MX-Functionsx 小节。

需要注意的是，原来的函数名 mcc_test 变成了 mlfMcc_test，这可以在 mylib.h 中看到。

在 VC++中新建工程，编写代码如下：

```
#include <stdio.h>
//包含头文件
#include "matrix.h"
#include "mylib.h"
int main(int argc, char* argv[])
{
double  x=0.04;
//创建mxArray型数据并对其赋值
mxArray *t = NULL;
mxArray *y = NULL;
//注册动态链接库
mylibInitialize();
//或者使用t=mxCreateDoubleScalar(x)
t = mxCreateDoubleMatrix(1,1,mxREAL);
y = mxCreateDoubleMatrix(1,1,mxREAL);
//或者使用MATLAB 数学库中的函数 mlfAssign()等
memcpy(mxGetPr(t),&x,sizeof(double));
//使用mylib.dll 中的函数
mlfMcc_test(1, &y, t);
getchar();
//销毁t，释放内存
mxDestroyArray(t);
//注销动态链接库
mylibTerminate();
return 0;
}
```

编译运行后输出如图 37-3 所示的结果。

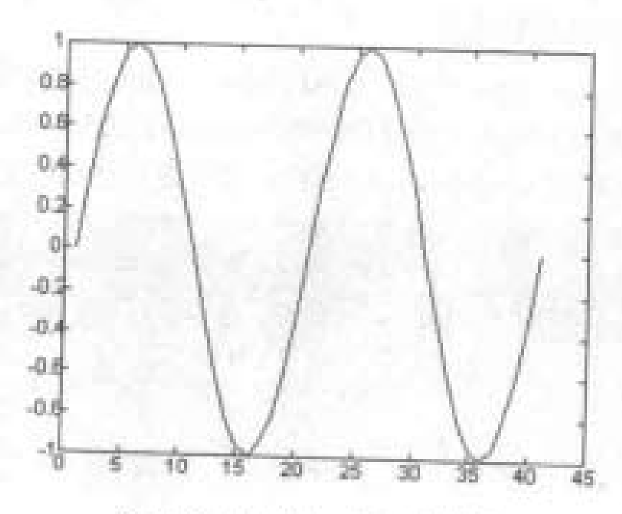


图 37-3 输出的 y=sin(x)的图形

程序完成之后可以进行打包发布。需要注意的是，发布的时候除了生成的 mylib.dll、FigureMenuBar.fig 和 FigureToolBar.fig，还需要加上 MATLAB 的一些动态链接库。这样发布的程序就可以完全脱离 MATLAB 的环境而独立运行了。

将<MATLAB>\extern\lib\win32\下的 mglinstaller.exe 解压缩，会得到 mglarchive.exe，再将 mglarchive.exe 解压缩，在生成的目录 bin\win32\下就是程序独立运行所需要的动态链接库文件了。

## 37.4 使用 COM

### 37.4.1 COM 简介

组件对象模型（COM，Component Object Model）是 Microsoft 提出的以组件为发布单元的软件开发技术。COM 具有与语言、平台无关的特性，可以方便软件的升级、定制与替换，是理想的软件应用方案。简言之，COM 是一种客户机/服务器标准，提供了一类应用程序接口，允许任何符合标准的程序访问。

MATLAB COM Builder1.0 于 2002 年 6 月在 MATLAB 6.5 版中推出，它编译的 COM 组件实现了标准的 IDispatch 接口，提供了对自动化（Automation）的支持。自动化对象公开了方法和属性。方法是指自动化对象提供的功能服务，类似于类的成员函数；属性是指自动化对象的数据特征，类似于成员变量。另外，为了实现组件对象与客户端的通信，增强程序的交互性，MATLAB 组件也支持事件。事件是回调函数，由客户端实现，事件发生时，组件会自动调用实现的客户端事件函数。

### 37.4.2 COM 的设置与创建

下面通过实例讲述通过 COM 实现 VC++和 MATLAB 的混合编程。

#### 1．COM Builder 编译环境设置

MATLAB COM Builder 在编译生成 COM 组件时，需要借助于外部的编译器，因此首先需要设置 MATLAB COM Builder 所使用的外部编译器，通过 mbuild-setup 完成编译环境的设置。在命令窗口中输入 mbuild-setup，并按照提示便可完成设置，代码如下所示。

```
    >> mbuild -setup
    Please choose your compiler for building standalone MATLAB applications:
    Would you like mbuild to locate installed compilers [y]/n? y
    Select a compiler:
    [1] Lcc C version 2.4 in D:\MATLAB7\sys\lcc
    [2] Microsoft Visual C/C++ version 6.0 in D:\Program Files\Microsoft Visual
Studio
    [0] None
    %选择编译器
    Compiler: 2
    Please verify your choices:
```





```
    Compiler: Microsoft Visual C/C++ 6.0
    Location: D:\Program Files\Microsoft Visual Studio
    Are these correct?([y]/n): y
    Try to update options file: C:\Documents and Settings\Administrator\
Application Data\MathWorks\MATLAB\R14\compopts.bat
    From template:          D:\MATLAB7\BIN\WIN32\mbuildopts\msvc60compp.bat
    Done . . .
    --> "D:\MATLAB7\bin\win32\mwregsvr D:\MATLAB7\bin\win32\mwcomutil.dll"
    DllRegisterServer in D:\MATLAB7\bin\win32\mwcomutil.dll succeeded
    --> "D:\MATLAB7\bin\win32\mwregsvr D:\MATLAB7\bin\win32\mwcommgr.dll"
    DllRegisterServer in D:\MATLAB7\bin\win32\mwcommgr.dll succeeded
```

MATLAB 自动注册 mwcomutil.dll 和 mwcommgr.dll，这两个 DLL 是 MATLAB COM Builder 生成的 COM 组件的基础，所有生成的 COM 组件都会使用到这两个 DLL。

### 2. 创建 COM 组件

创建 COM 组件的基本步骤如下。

步骤 1：首先在 MATLAB 的命令窗口中输入命令 comtool，启动 COM Builder 的图形用户界面如图 37-4 所示。

步骤 2：使用 File->New Project…建立新的工程，会出现如图 37-5 所示的工程设置窗口。




图 37-4 MATLAB COM Builder 主窗口

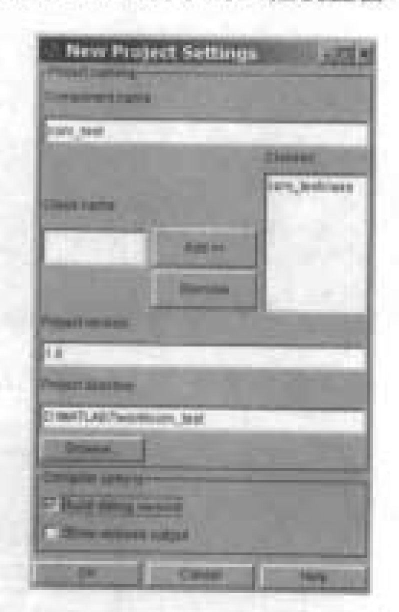



图 37-5 工程设置窗口

在 Component name 中输入要生成的 COM 组件的名字，本例中输入名字 com_test，最后生成的 COM 组件的名字就是这个名字加上版本信息。通过 Add>>和 Remove 按钮可以为这个 COM 组件添加和删除类，一个 COM 组件可以包含许多类，在 Project Setting 里面能实现类的添加。单击“OK”，进入如图 37-6 所示的对类操作的窗口。

步骤 3：在图 37-6 的窗口中，对于每个类，可以为其添加方法（methods），属性（Properties）和事件（events），下面分别进行讲述。

（1）添加类的方法（methods）

给类添加方法非常的简单，操作如下：先用鼠标选中所要进行操作的类，然后使用菜





单 Project->Add File…或者按钮 Add File 添加现成的（预先编写好的）文件就可以了。注意.m文件不能是脚本文件，只能是函数文件。



图 37-6 对类进行操作的窗口

本例中，为这个类添加一个简单的方法，测试一下函数 Plot。文件代码如下：

```
function plot_test
t=0:0.001:0.04;
x=cos(2*pi*50*t);
plot(t,x,'r--');
grid on;
title('COM 组件绘图例子');
```

这样就为类添加了一个方法，如图 37-7 所示。



图 37-7 添加类方法的窗口



（2）添加类的属性（Properties）

MATLAB COM builder 自动把形成类的方法的.m 函数中所包含的全局变量转换为类的属性，.m 中的全局变量是通过关键字 global 定义的。本例中，再为这个类添加如下方法。

```
function result=property_test()
global A;
if(isempty(A))
result=0;
return;
end
result=det(A);
```

经过编译，A 就变成了类 com_test 的一个属性。

（3）添加类的事件（events）

为类添加事件只需要用到语法%#event 就可以了。本例将下面的函数加入到类中，经过编译就会形成这个类的一个事件。函数文件为：

```
function event_test(i)
%#event
i
```

在 MATLAB 的环境下执行的时候，%#event 语句就被当作是注释，但在 COM Builder 中就起作用了。

步骤4：所有函数添加完毕之后，就可以使用菜单 Build->COM Object 或者按钮 Build 进行编译了，如图 37-8 所示。

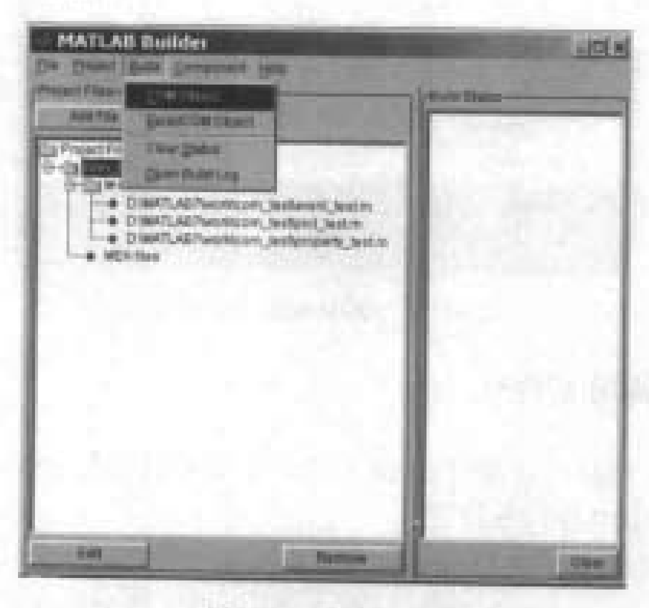



图 37-8　编译 COM

编译过程如图 37-9 所示，图中右侧的窗口显示编译过程的信息。

编译完毕之后，在工程文件夹下生成两个文件夹：一个是 src，里面存放的是一些中间文件；一个是 distrib，里面就是创建的 COM 的文件。

在编译完毕之后，MATLAB 会自动组成生成的 component，可以使用菜单



Component->Component Info…将系统注册表里面有关于 com_test_1_0.dll 的详细信息调出来看一看有关生成的 component 的信息，如图 37-10 所示。



图 37-9　编译 COM 的过程窗口



图 37-10　COM 信息显示窗口

### 37.4.3　VC++中调用 COM

用 VC++实现客户端程序，调用 COM 组件，主要步骤如下所示。

步骤 1：对 VC++IDE 作如下设置。

在 VC++IDE 中选择 Tools->Options->Directories。

在 Show directories for:中选择 Include files，添加如下两个目录：

<MATLAB>\extern\include\

<MATLAB>\extern\include\cpp

在 Show directories for:中选择 Library files，添加如下两个目录：

<MATLAB>\extern\lib\win32

<MATLAB>\extern\lib\win32\microsof\msvc6





这里假设<MATLAB>为 MATLAB 的安装目录。

步骤 2：通过 ClassWizard→AddClass→From a type Library 将上面编译好的组件 com_test_1_0.dll 加入工程，在 ClassView 中多了两个类 IMATLABClass 与 IMATLABClass Events，均派生自 COleDispatchDriver，它是 MFC 提供的一个封装类，主要用于自动化对象的客户调用 IDispatch::Invoke()函数。

步骤 3：在 BOOL CCom_test4App::InitInstance()中添加 AfxOleInit()，类 CCom_test4App 的构造函数中添加 EnableAutomation()。在 com_test4Dlg.cpp 添加#include "com_test_1_0.h"，参照 com_test_idl_i.c 文件中的定义，在 com_test4Dlg.cpp 中添加如下代码：

```
    #ifndef __IID_DEFINED__
    #define __IID_DEFINED__
    typedef struct _IID
    {
        unsigned long x;
        unsigned short s1;
        unsigned short s2;
        unsigned char  c[8];
    } IID;
    #endif // __IID_DEFINED__

    #ifndef CLSID_DEFINED
    #define CLSID_DEFINED
    typedef IID CLSID;
    #endif // CLSID_DEFINED
    const IID IID_Icom_testclass = {0xEE365135,0xB390,0x40B0,{0xAD,0xBD,0xE9,
0x01,0xEF,0xEA,0x4B,0x6F}};
    const IID LIBID_com_test = {0xACE823EF,0x2AA8,0x4293,{0xA7,0x29,0xF4,0x51,
0xC5,0x00,0x4E,0xB2}};
    const IID DIID__Icom_testclassEvents = {0xFBEEFDC1,0xDE19,0x4F86,{0x9B,
0xDA,0x68,0x1A,0x39,0x0F,0x8F,0x99}};
    const CLSID CLSID_com_testclass = {0xD42C1BE6,0xCBBF,0x4B2B,{0x96,0x96,
0xF9,0xCD,0x81,0x68,0x92,0xAC}};
```

步骤 4：在对话框“确定”按钮的响应函数 IDOK()中添加代码：

```
    Icom_testclass *test = new Icom_testclass();
    test->CreateDispatch(CLSID_com_testclass);
    test->plot_test();
```

步骤 5：编译、运行程序，单击对话框上的“确定”按钮，输出如图 37-11 所示的图。

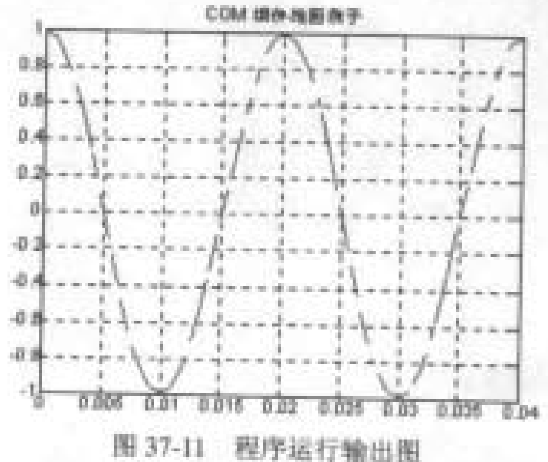



图 37-11　程序运行输出图





上面通过一个简单的实例讲述了 VC++和 MATLAB 通过 COM 混合编程的过程。

## 37.5 小结

本章通过实例讲述了常用的 MATLAB 和 VC++混合编程的三种方法：使用引擎、使用 mcc 编译器和使用 COM。通过实例的具体操作，读者可掌握这些基本方法的应用，并能扩展到 MATLAB 与其他高级语言的混合编程。

